DOUBLY STOCHASTIC MATRICES AND THE ASSIGNMENT PROBLEM

by

Maria Sivesind Mehlum

Thesis

for the degree of

Master of Science

(Master i Matematikk)

Desember 2012

Faculty of Mathematics and Natural Sciences
University of Oslo

Acknowledgements

First of all, I would like to thank my supervisor, Geir Dahl for valuable guidance and advice, for helping me see new expansions and for believing I could do this when I didn't do it myself. I would also like to thank my fellow "LAP Real" students, both the original and current, for long study hours and lunches filled with discussions, frustration and laughter. Further, I would like to thank my mum and dad for always supporting me, and for all their love and care through the years. Finally, I would like to thank everyone who has been there for me over the past 17 weeks with encouragement and motivational distractions which has made this time enjoyable.

Contents

Chapter 1

Introduction

In linear optimization we are considering problems that can be presented as linear programming problems, which will be defined in chapter 2. The topic of this thesis will be the class of doubly stochastic matrices, and when they are related to linear optimization problems, the restrictions they apply will give rise to some special kinds of problems, among them the assignment problem, which will be the main problem of this thesis and will be defined and presented in chapter 4 together with some theoretical results, three algorithms for solving such problems and some applications and examples. We will use both matrix and graph notation, and therefore, chapter 2 gives a short presentation of these approaches. The doubly stochastic matrices will be properly defined in chapter 3 together with some related mathematical concepts. The set of such $n \times n$ matrices, which is called the Birkhoff polytope and will be denoted Ω_n, is closely related to the main theorem of this thesis, namely the Birkhoff and von Neumann theorem, which will be presented in chapter 5 together with two different proofs, the first one based on matrix notation whereas the second proof is taking a graph approach. The secondary problem of this thesis, is a version of the assignment problem with some additional constraints and will be presented and discussed in chapeter 6. We will also explore some variations of this problem that are more or less restricted than the original. The implementations of the presented algorithms used in order to solve the problems in the examples in this thesis will be presented in appendix A as program codes in Matlab or OPL-CPLEX. Because this thesis is part of my teacher education, I will in conclusion present some didactive reflections of a possible inclusion of the assignment problem into the math curriculum in the Norwegian high school (chapter 7).

The theoretical background of this thesis is based on the selection of books and articles presented in the bibliography. The introduction to each chapter

will specify the sources of the upcoming theory in more detail. However, my main contribution has been to present the two linear programming problems mentioned above in relation to doubly stochastic matrices with both a matrix and a graph approach. This involves presenting appropriate theoretical results and algorithms supported by relevant applications and examples. Although all of the proofs presented in this thesis will be my own representations, some are influenced by existing proofs to a greater extent. Therefore, I want to highlight the following proofs as being more or less my own production throughout.

1. the proof of Proposition 3 in chapter 2.2
2. the proof of Proposition 4 in chapter 3.1
3. the proof of Corollary 6 in chapter 3.3
4. the proof of Lemma 7 in chapter 3.3
5. the proof of Theorem 9 in chapter 4.2
6. the proof of Proposition 12 in chapter 5.2
7. the proof of Theorem 13 in chapter 5.2
8. the second proof of Theorem 11 in chapter 5

Also, all examples, including program codes, except example 4 are my own creations.

Chapter 2

Theoretical Background

This chapter will give an introduction to some basic concepts of this thesis. We will start with a short, but sufficient presentation of the Linear Programming problem, onwards denoted the LP problem with matrix notation which will be based on the theory in the book by Vanderbei [10]. Then, we will continue by presenting some properties of the solution space of such problems, for which convexity theory will be central, based on *An Introduction to Convexity* by Dahl [6]. Finally, we will give a short introduction to graph theory, which will be based on Vanderbei's book [10] and a compendium by Dahl [5], and state the LP problem with this notation as well.

2.1 Linear optimization

In linear optimization problems the goal is to maximize or minimize the value of a given linear function of a variable $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, which is called the decision variable, due to some constraints given as linear inequalities or equalities of x. This will be the primal LP problem, which is on the form

$$\begin{array}{ll} \text{Maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{array} \tag{2.1}$$

for a maximization problem. We want to find the maximum value of $c^T x$, called the objective function satisfying all the constraints given in the linear system $Ax \leq b$, where A is an $m \times n$ matrix. Therefore, we get m constraints in total. Usually, there is more than one x satisfying these constraints. Together they will constitute a set called the feasible solutions of the problem. If a given x in this set also maximizes $c^T x$, we say that this solution is optimal. Figure 2.1 shows an example of an LP problem in $\mathbb{R}^2$.

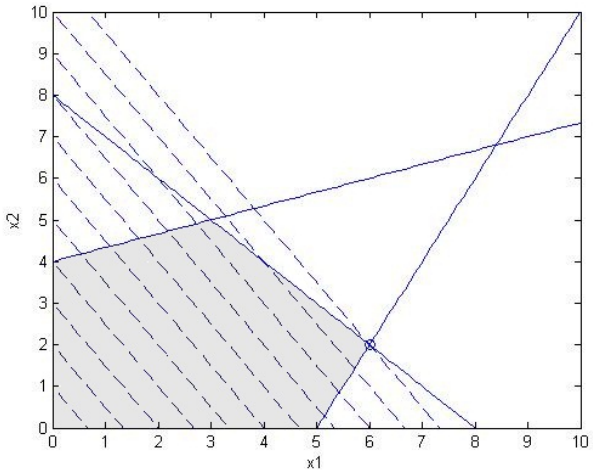


Figure 2.1: LP problem - an example

The linear functions enveloping the marked area represent the constraints of the problem, which means that the feasible solutions of the problem must be contained in this area. The parallel dashed lines represent some feasible solutions of this problem. We observe that the value of the objective function grows as the dashed lines are moved to the right. Since it cannot exceed the given boundaries, the maximum will be achieved when $x = 6$ and $y = 2$, which is one of the vertices of the feasible area and marked by a circle, i.e. this is the optimal solution of the problem. In order to solve problems of higher dimensions, algorithmic methods are necessary. The Simplex method is probably the most famous such algorithm, for which the reader is referred to Vanderbei [10] for a thorough description. The program OPL-CPLEX implements this algorithm computationally and will be used in solving some of the examples presented later.

In this thesis, the LP problems will have decision variables $X \in \mathbb{R}^{n\times n}$, i.e. X is an $n \times n$ matrix, and we get problems on the form

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ij}x_{ij} \leq a_j & \text{for } j = 1,\ldots,n \\ & \sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i & \text{for } i = 1,\ldots,n \\ & x_{ij} \geq 0 & \text{for } i,j = 1,\ldots,n \end{aligned} \tag{2.2}$$

or alternatively

$$\begin{array}{ll} \text{Maximize} & \langle C, X\rangle \\ \text{subject to} & \langle A, X\rangle \leq a \\ & \langle A, X\rangle \leq b \\ & X \geq 0 \end{array} \tag{2.3}$$

where A and C also will be $n \times n$ matrices, and the inner product is defined by $\langle A, B\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij}$. The dual problem will be

$$\begin{array}{ll} \text{Minimize} & a^T u + b^T v \\ \text{subject to} & u^T A + v^T A \leq C \\ & u, v \in \mathbb{R}^n \end{array} \tag{2.4}$$

and will be relevant later on together with the relationship between the primal and the dual problem

$$x_{ij}(c_{ij} - u_i - v_j) = 0 \tag{2.5}$$

which is called the complementary slackness conditions.

2.2 Convexity

We say that a set C is convex if for any pair of points in the set, $x_1, x_2 \in C$, the line segment, $(1-\lambda)x_1 + \lambda x_2$, between these points also lies in C for $0 \leq \lambda \leq 1$. Thus, we can think of a convex set as a set with no holes in it. Figure 2.2 shows some examples of convex sets, whereas figure 2.3 shows some nonconvex sets.

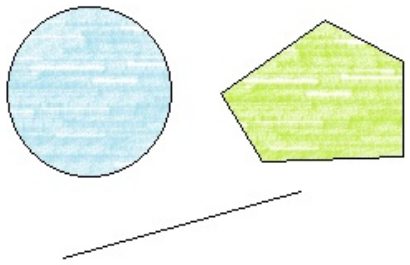


Figure 2.2: Three convex sets

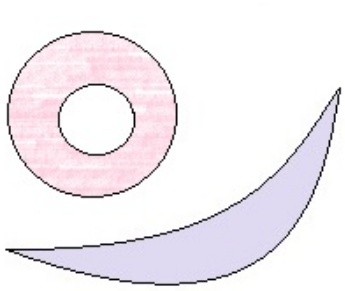


Figure 2.3: Two sets that are not convex

Further, we define a convex combination to be a vector

$$x = \sum_{i=1}^{t} \lambda_i x_i$$

where $x_1, ..., x_t \in \mathbb{R}^n$, $\lambda_i \geq 0$ for $j = 1, \ldots, t$ and $\sum_{i=1}^{t} \lambda_i = 1$. Then, we have

Proposition 1. *A set $C \subseteq \mathbb{R}^n$ is convex if and only if it contains all convex combinations of its points.*

The following proof is based on the proof given in the compendium by Dahl [6].

Proof. $\Rightarrow$: Let $C = \{x_1, \ldots, x_n\}$ and suppose C is convex. This means that C contains all line segments of any pair of points in C, i.e. $\lambda x_1 + (1-\lambda)x_2 \in C$ for any $x_1, x_2 \in C$ and $0 \leq \lambda \leq 1$, which is a convex combination. Hence, C is containing all convex combination of any 2 of its points. We continue the proof by induction, assuming that C contains all convex combinations of $n-1$ of its points, i.e. we have $\sum_{i=1}^{n-1} \lambda_i x_i \in C$ for $x_1, \ldots, x_{n-1} \in C$ and $\sum_{i=1}^{n-1} \lambda_i = 1$. Then, we must show that this also holds for n of the points. We consider the convex combination

$$\sum_{i=1}^{n} \lambda_i x_i$$

where $x_1, \ldots, x_n \in C$ and $\sum_{i=1}^{n} \lambda_i = 1$, which we must show that lies in C.

$$\sum_{i=1}^{n} \lambda_i x_i = \sum_{i=1}^{n-1} \lambda_i x_i + \lambda_n x_n = \frac{1}{1-\lambda_n} \sum_{i=1}^{n-1} \frac{1}{1-\lambda_n} \lambda_i x_i + \lambda_n x_n$$

which is a convex combination of the two points x_n and $y = \sum_{i=1}^{n-1} \frac{1}{1-\lambda_n}\lambda_i x_i$. Because $\sum_{i=1}^{n-1} \frac{1}{1-\lambda_n}\lambda_i = 1$, y is a convex combination of $n-1$ of the points in C. Since all convex combinations of $n-1$ of the points in C lie in C by assumption, and x_n obviously lies in C, this convex combination must also lie in C. Hence, C contains all convex combinations of n of the points in C and therefore all convex combinations of any of the points in C must be contained in C.

$\Leftarrow$: Now we assume that C contains all convex combinations of its points. Since the line segment between any two points $x_1, x_2 \in C$ can be written $\lambda x_1 + (1-\lambda)x_2$, for which we obviously have $\lambda + (1-\lambda) = 1$, this is a convex combination of x_1 and x_2, and C must therefore be convex. □

Convexity theory can be used to describe many mathematical concepts. Among them, the set of feasible solutions of LP problems presented in the previous section, which is in fact a convex set. More precisely, this set is called a polyhedron, which is defined to be the solution set of any linear system and is therefore on the form

$$\{x \in \mathbb{R}^n : Ax \leq b\} \text{ for } A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$$

Another important concept is the convex hull, which is defined to be the set of all convex combinations of the points in a given set S. We denote this set as $\text{conv}(S)$, and it will be convex no matter what S is due to proposition 1. We have in fact that

Proposition 2. *If we let $S \subseteq \mathbb{R}^n$. Then, conv(S) is equal to the intersection of all convex sets containing S. Thus, conv(S) is the smallest convex set containing S.*

The following proof is based on the proof given in the compendium by Dahl [6].

Proof. We know that $\text{conv}(S)$ consists of elements on the form $\sum_{i=1}^{t} \lambda_i x_i$ for $x_1, \ldots, x_n \in S$ and where $\sum_{i=1}^{t} \lambda_i = 1$. We let H be the intersection of all convex sets containing S and must show that $\text{conv}(S) = H$.

H $\subseteq$ conv(S): We know that $S \subseteq \text{conv}(S)$. Therefore, we must also have $H \subseteq \text{conv}(S)$.

H $\supseteq$ conv(S): We let C_i be convex sets such that $S \subseteq C_i$, i.e.
$S \subseteq \cap C_i = H$. Then, we also have that $\cap C_i$ is convex since the intersection of convex sets is convex. Since $H \supseteq S$ and H is convex, H must contain all convex combinations of S, hence $\text{conv}(S) \subseteq H$.

By this, $H = \text{conv}(S)$, which is what we wanted to prove. □

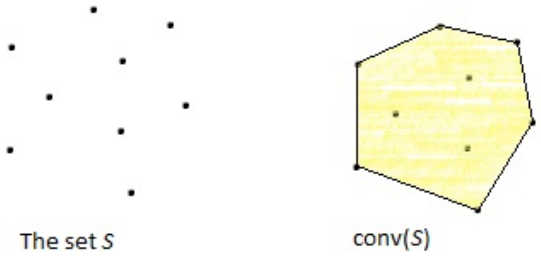


Figure 2.4: The convex hull

If S is a finite set of points, we say that conv(S) is a polytope. A polytope is also defined to be a bounded polyhedron and is important in LP theory because the set of feasible solutions of most LP problems are polytopes. In fact, the feasible set of an LP problem is a polytope if it is bounded. In figure 2.1, the marked area is a polyope in $\mathbb{R}^2$. When we have points $x \in C$, which cannot be written as the mid-point between two other points in the set, i.e. $x_1, x_2 \in C$ such that $x = \frac{1}{2}x_1 + \frac{1}{2}x_2$ do not exist, we say that x is an extreme points of the set. Graphically, this can be seen as the vertices of the polytope. Therefore, we can also say that a polytope is the convex hull of its extreme points. Figure 2.4 shows an example of the convex hull of a finite set of points S. As we can see, it is determined by the points becoming the vertices of the polytope, i.e. the extreme points of conv(S).

We end this section by considering the convex hull of a set of points, which are what we call affinely independent. The set $\{x_1, \ldots, x_t\} \in \mathbb{R}$ is affinely independent if $\sum_{j=1}^{t} \lambda_j x_j = 0$ and $\sum_{j=1}^{t} \lambda_j = 0$ imply that $\lambda_1 = \ldots = \lambda_t = 0$. The convex hull of this set is called a simplex, for which $\{x_1, \ldots, x_t\}$ will be the vertices. For any selection of $k+1$ of these vertices we get a k-simplex, which will be a k-dimensional polytope. Hence, $dim(S_n) = n - 1$ where S_n is the set of n affinely independent vectors. Another important property of simplices is given in the following proposition.

Proposition 3. *Let the vectors $x_1, \ldots, x_t \in \mathbb{R}^n$ be affienly independent and consider the simplex $S = conv(\{x_1, \ldots, x_t\})$ generated by these vectors. Then each point in S has an unique representation as a convex combination of $x_1, \ldots, x_t$.*

Proof. Let $y \in \text{conv}(\{x_1, \ldots, x_t\})$. Then, there exists a convex combination

of y of the vectors $x_1, \ldots, x_t$ by the definition of convex hulls. We must show that this convex combination is unique. Assume for contradiction that the convex combination is not unique, i.e. we have

$$y = \sum_{j=1}^{t} \lambda_j x_j = \sum_{j=1}^{t} \mu_j x_j$$

where $\sum_{j=1}^{t} \lambda_j = \sum_{j=1}^{t} \mu_j = 1$ and $\lambda_j \neq \mu_j$ for $j = 1, \ldots, t$. This means that we can write

$$(\lambda_k - \mu_k)x_k = (\mu_1 - \lambda_1)x_1 + \ldots + (\mu_{k-1} - \lambda_{k-1})x_{k-1} + (\mu_{k+1} - \lambda_{k+1})x_{k+1} + \ldots + (\mu_t - \lambda_t)x_t$$

$$\Downarrow$$

$$x_k = \sum_{j=1, j\neq k}^{t} \frac{\mu_j - \lambda_j}{\lambda_k - \mu_k} x_k$$

where

$$\begin{aligned} \sum_{j=1, j\neq k}^{t} \frac{\mu_j - \lambda_j}{\lambda_k - \mu_k} &= \frac{1}{\lambda_k - \mu_k} \sum_{j=1, j\neq k}^{t} (\mu_j - \lambda_j) \\ &= \frac{1}{\lambda_k - \mu_k}((1 - \mu_k) - (1 - \lambda_k)) = \frac{1}{\lambda_k - \mu_k}(\lambda_k - \mu_k) = 1 \end{aligned}$$

Hence, we have written x_k as a convex combination of the other vertices of S, which contradicts the fact that $\{x_1, \ldots, x_t\}$ is an affinely independent set. Therefore, we must have $\lambda_j = \mu_j$ for $j = 1, \ldots, t$, which means that the convex combination is unique. □

2.3 Network - Flow

We have already introduced the LP problem with matrix notation, see equations (2.1) and (2.2). Alternatively, such problems can be presented by graph theory by expressing the matrices as certain graphs. In this section we will present some basic properties of network-flow theory, and most importantly, express the LP problem in graph notation.

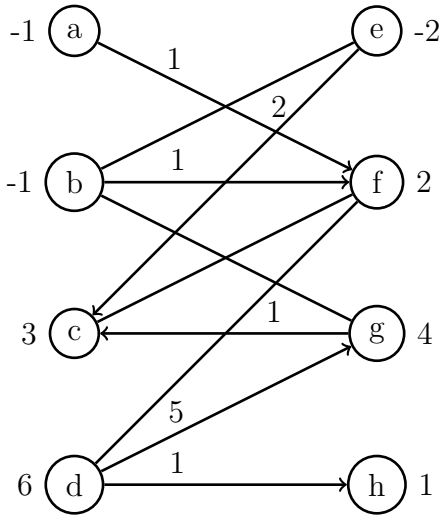
In network-flow theory we define a graph $G = (U; E)$ to consist of the vertices $v \in U$ and the edge set E, which consist of elements on the form $e = (v, w)$ connecting pairs of distinct elements of U, e.g. the vertices v and w. If the graph is directed we say that the edge goes from v to w. Hence, we have to differ between edges entering and leaving some vertex v. Therefore, we define

$\delta^+(v) = \{e \in E : e = (v, w) \text{ for some vertex } w \in U\}$: the set of edges leaving v
$\delta^-(v) = \{e \in E : e = (u, v) \text{ for some vertex } u \in U\}$: the set of edges entering v

Transportation problems, which are a common application of the network-flow theory, are problems where we want items to be transported from one place to another. This corresponds to having an edge between two vertices with an assigned value corresponding to the amount transported the given distance. This value is called the flow of this edge and is given by a function $x : E \to \mathbb{R}$ such that the value $x(e)$ is the flow in edge e, which is usually denoted x_{uv} as the flow from u to v. Then, $\sum_{\delta^+(v)} x_{uv}$ is the sum of all flows leaving v and $\sum_{\delta^-(v)} x_{vw}$ is the sum of all flows entering v. The difference

$$\sum_{\delta^+(v)} x_{uv} - \sum_{\delta^-(v)} x_{vw} = -b(v)$$

is called divergence, and the sign of this value decides whether v is a demand or a supply vertex.



In the graph above, the divergences are shown as the numbers next to the vertices, e.g. $b(a) = -1$, which means that a is a supply vertex, whereas $b(f) = 2$ means that f is a demand vertex. All the edges in the graph denotes where we possibly can have any edge flows. Here, the numbers above some of the edges represent a possible feasible solution. This can for instance be found by identifying a leaf node, i.e. a vertex with only one adjacent edge, assign the appropriate value to the connected edge and proceed. For each edge, we also add costs, c_{uv}, which denotes the unit cost of a flow between u and v. This results in a maximization/minimization problem where optimality becomes an issue.

We now continue by presenting some graph related concepts. A path is defined to be a series of vertices where each adjacent pair of vertices is connected

by an edge in the graph. If there exists a path connecting all the vertices, we say that the network is connected. Furthermore, if the path begins and ends at the same vertex, it is a cycle. If the network contains no cycles, it is acyclic. By this we can define a tree to be a network that is acyclic and connected, and a forest to be a group of trees, naturally.

So far, we have considered our vertex set to be a random collection of vertices. From now on our vertex set, U, will be a set consisting of two disjoint sets, V and W, such that $U = V \cup W$ and $V \cap W = \emptyset$, and where all edges in the graph connects a vertex in V with a vertex in W. We say $G = (V, W; E)$ is a bipartite graph. The graph above is an example of a bipartite graph. We let $|V| = |W| = n$ such that we get an $n \times n$ problem. The flows of this problem are presented in an $n \times n$ matrix X, where the rows represents vertices in V and the columns represent vertices in W. Now, we can write the minimization LP problem as a minimum cost network flow problem

$$\begin{aligned} \text{Minimize} \quad & \sum_{(u,v)\in E} c_{uv}x_{uv} \\ \text{Subject to} \quad & \sum_{u:(u,v)\in E} x_{uv} - \sum_{w:(v,w)\in E} x_{vw} = -b(v) \quad (v \in U) \\ & 0 \leq x_{uv} \leq a_{uv} \qquad\qquad\qquad\qquad\qquad ((u,v) \in E) \end{aligned}$$

These two ways of presenting LP problems will both be used in the rest of this thesis depending on what we are presenting.

Chapter 3

Doubly stochastic matrices

A square matrix is said to be doubly stochastic if its elements are non-negative and all row sums and column sums are equal one. In this chapter, such matrices and some of their properties will be defined and described. Then, a closely related concept, majorization, will be presented and linked to these matrices, which will yield some further results. The theory of this chapter will be based on the book by Marshall and Olkin [9] and the article by Burkhard and Çela [3], both defining doubly stochastic matrices in the following way.

Definition. *An $n \times n$ matrix $A = (a_{ij})$ is doubly stochastic if*

(i) $a_{ij} \geq 0$ *for* $i, j = 1, ..., n$

(ii) $\sum_{i=1}^{n} a_{ij} = 1, \quad j = 1, ..., n; \qquad \sum_{j=1}^{n} a_{ij} = 1, \quad i = 1, ..., n$

3.1 The Birkhoff Polytope

The set of all $n \times n$ matrices satisfying the constraints in the definition above is called the Birkhoff polytope, denoted by Ω_n, and has the following property

Proposition 4. *The set of all $n \times n$ doubly stochastic matrices, Ω_n, is convex.*

Proof. Let $A, B \in \Omega_n$ and $0 \leq \lambda \leq 1$. Then, we must show that $\lambda A + (1 - \lambda)B \in \Omega_n$.

$$\lambda A + (1-\lambda)B = \begin{pmatrix} \lambda a_{11} & \dots & \lambda a_{1n} \\ \vdots & & \vdots \\ \lambda a_{n1} & \dots & \lambda a_{nn} \end{pmatrix} + \begin{pmatrix} (1-\lambda) b_{11} & \dots & (1-\lambda) b_{1n} \\ \vdots & & \vdots \\ (1-\lambda) b_{n1} & \dots & (1-\lambda) b_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} \lambda a_{11} + (1-\lambda)b_{11} & \dots & \lambda a_{1n} + (1-\lambda)b_{1n} \\ \vdots & & \vdots \\ \lambda a_{n1} + (1-\lambda)b_{n1} & \dots & \lambda a_{nn} + (1-\lambda)b_{nn} \end{pmatrix}$$

with row sums

$$\begin{aligned} \sum_{j=1}^{n} (\lambda a_{ij} + (1-\lambda)b_{ij}) &= \lambda \sum_{j=1}^{n} a_{ij} + (1-\lambda) \sum_{j=1}^{n} b_{ij} \\ &= \lambda + 1 - \lambda = 1 \quad \text{for } i = 1, \dots, n \end{aligned}$$

and column sums

$$\begin{aligned} \sum_{i=1}^{n} (\lambda a_{ij} + (1-\lambda)b_{ij}) &= \lambda \sum_{i=1}^{n} a_{ij} + (1-\lambda) \sum_{i=1}^{n} b_{ij} \\ &= \lambda + 1 - \lambda = 1 \quad \text{for } j = 1, \dots, n \end{aligned}$$

Hence, $\lambda A + (1-\lambda)B \in \Omega_n$. □

A special subgroup of these matrices is the permutation matrices, which are $n \times n$ matrices where each row and each column only has one nonzero entry, which therefore is equal to one. There are $n!$ such $n \times n$ matrices, and we denote the set of these matrices as P_n. Both Ω_n and P_n will be essential for this thesis and will later on be linked together by the main theorem of this thesis. Some further properties of Ω_n are given in the following theorem

Theorem 5. (Birkhoff, 1946)

(i) *For $A \in \Omega_n$, $dim(A) = (n-1)^2$*

(ii) *Each of the $n!$ vertices of Ω_n coincides with $\sum_{k=0}^{n-1} \binom{n}{k}(n-k-1)!$ different edges*

(iii) *Any two vertices are joined by a path with at most two edges, i.e. the diameter of Ω_n is 2*

(iv) *Ω_n is Hamiltonian, where Hamiltonian means that the polytope has a closed path along its edges which visits each vertex of the polytope just once*

3.2 Examples

To get a better understanding of doubly stochastic matrices and their importance for this thesis, we present some examples of doubly stochastic matrices of lower dimension.

Example 1. *We start by the case when* ***n=1****. Then, we only have the element* $A = 1$*. Even though this is an integer rather than a matrix, it clearly satisfies the requirements for being doubly stochastic.*

The next examples are more interesting.

Example 2. *When* ***n=2*** *we get matrices of the following form*

$$A = \begin{pmatrix} \alpha & 1-\alpha \\ 1-\alpha & \alpha \end{pmatrix}$$

where α *is any number* $0 \leq \alpha \leq 1$*. We observe that when the first entry,* α*, has been chosen, the rest of the entries of the matrix are given. Hence,* $dim(A) = 1$*, which satisfies theorem 5. For a transportation problem, we can interpret this as a case where we have two factories and two customers. If the first factory decides to transport a certain percentage,* α*, of its items to the first customer, the other factory has to transport a percentage* $1 - \alpha$ *of its items to the same customer, i.e. the rest of what that customer wanted to buy. Also, the first factory has to transport a percentage* $1-\alpha$ *of its items to the second customer, whereas the second factory transports a percentage* α *of its items to this customer, which again corresponds to the rest of what that customer wanted to buy.*

Example 3. *For* ***n=3****, the form of the matrices is even more complex.*

$$A = \begin{pmatrix} \alpha_1 & \alpha_2 & 1-\alpha_1-\alpha_2 \\ \alpha_3 & \alpha_4 & 1-\alpha_3-\alpha_4 \\ 1-\alpha_1-\alpha_3 & 1-\alpha_2-\alpha_4 & \alpha_1+\alpha_2+\alpha_3+\alpha_4-1 \end{pmatrix}$$

Here, we have four free variables $\alpha_1, \alpha_2, \alpha_3, \alpha_4$*, where* $0 \leq \alpha_i \leq 1$ *for* $i = 1, \ldots, 4$*, which must satisfy*

$$\begin{aligned} \alpha_1 + \alpha_2 &\leq 1 \\ \alpha_1 + \alpha_3 &\leq 1 \\ \alpha_2 + \alpha_4 &\leq 1 \\ \alpha_3 + \alpha_4 &\leq 1 \end{aligned}$$

These variables can be positioned differently than presented above, the point is that when four entries are decided, the rest are given. Because there are four free variables, we have $dim(A) = 4$*.*

For the next property of doubly stochastic matrices we need to introduce a new concept, majorization.

3.3 Majorization

The concept of majorization describes how two vectors are related to each other in a certain way. For vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ we let $x_{[j]}$ and $y_{[j]}$ denote the jth biggest element of x and y respectively. Then, we can define majorization in the following way.

Definition. *For $x, y \in \mathbb{R}^n$, x is majorized by y, denoted $x \prec y$, if*

(i) $\sum_{i=1}^{k} x_{[i]} \leq \sum_{i=1}^{k} y_{[i]}, \quad k = 1, \ldots, n-1$

(ii) $\sum_{i=1}^{n} x_{[i]} = \sum_{i=1}^{n} y_{[i]}$

We can describe this by saying "x is less spread out than y". To illustrate this concept, we will present an example which is related to income distribution in populations. This example is based on the example given in the book by Marshall and Olkin [9].

Example 4. *We have two populations, X and Y, each having n members. The total wealth of all the members in each group is denoted by T, but the incomes are not equally distributed. We order the members of population X and Y from the poorest to the richest, such that $x_{[i]}$ and $y_{[i]}$ are the ith poorest persons in population X and Y respectively. Now, we define the relative wealth of the ith individual to be the sum of the ith poorest members of a population divided by T. We denote these values by $S_{x[i]}$ and $S_{y[i]}$. We also define $n_{[i]}$ to be the proportion of the population having an income of $x_{[i]}$ ($y_{[i]}$) or less, such that $n_{[n]}$ denotes the whole population. Then, by plotting $(n_{[i]}, S_{x[i]})$ and $(n_{[i]}, S_{y[i]})$ in the same coordinate system, we can observe how the wealth distributions in the populations X and Y are related, see figure 3.1. The straight line describes a population where all incomes are equal. For all populations, the graphs will start in origo and end in (1,1), but the shapes will be different. In the figure, we observe that the graph for population X is less convex than the graph for population Y. We can then conclude that the income in population X is more evenly distributed than in population Y, i.e. x is less spread out than y. We also observe that*

$$\sum_{i=1}^{k} y_i \leq \sum_{i=1}^{k} x_i \quad \text{for } k = 1, \ldots, n-1$$

In this example, x and y are ordered from the poorest to the richest member of the populations instead of the other way around as in the definition of

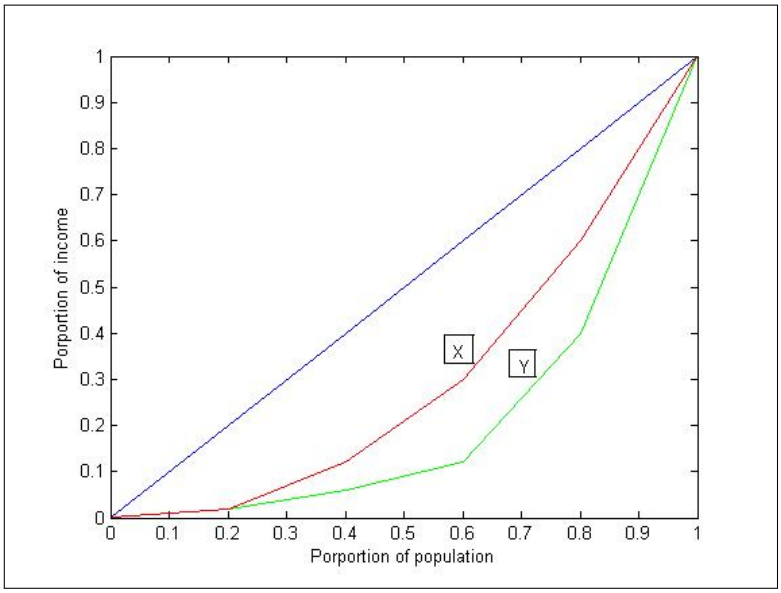


Figure 3.1: Wealth distributions in population X and Y

majorization. Therefore, the inequality sign has been reversed. If we let x^* *and* y^* *denote the vectors* x *and* y *ordered as in the definition, we can equivalently write*

$$\sum_{i=1}^{k} x_i^* \leq \sum_{i=1}^{k} y_i^* \quad \textit{for } k = 1, \ldots, n-1$$

Also,

$$\sum_{i=1}^{n} x_i^* = \sum_{i=1}^{n} y_i^* = T$$

by defition. Hence, $x \prec y$.

Such income distributions can be used in order to give a measure of the wealth inequality in populations. The Gini index, which was developed by the Italian sociologist and statistician Corrado Gini in 1912 [1], is such a measure. The index is determined by first presenting the income distributions of the populations as in the graphs in figure 3.1. The straight line is called the equidistribution line, A is the area between the equidistribution line and the distribution curve of a population, and the total area under the equidistribution line is denoted $A + B$. Then, we can calculate this ratio which is called the Gini index, G,

$$G = \frac{A}{A + B}$$

Calculations show that we can also write

$$G = 1 - 2B = 2A$$

Hence, the Gini index for a population equals twice the size of the area between the equidistribution line and the distribution curve of that population. We observe that $0 \leq G \leq \frac{n-1}{n} \to 1$ as $n \to \infty$, where $G = 0$ represents a population where all incomes are equal, and $G = 1$ corresponds to a population where all incomes are zero except one which is equal to the total income of that population, meaning that one person owns all of the income.

Another property of the Gini Index is that if we multiply each income by some factor α, say each individual in a population gets a certain percentage pay rise, the Gini index remains the same. Whereas if this pay rise was the same amount of money for every individual, i.e. we add a constant value c to each income, the Gini index will increase. Hence, the Gini index is scale invariant, but not translation invariant. In order to decrease the Gini index of a population for which the total income is given, an income redistribution where the income is redistributed from the richer to the poorer individuals of the population, is necessary.

Because the vectors of our data sets are ordered decreasingly by the definition of majorization, the original order of the elements has no significance. Therefore, we can put the elements of a set in whatever order we want without changing the majorization property of the sets. Hence, $y \prec Py$ for any permutation matrix P. With this in mind we will consider the set we get by taking all kinds of permutations of y, which will give us another property.

Corollary 6. *The set $\{x : x \prec y\}$ is the convex hull of points obtained by permuting the components of y.*

Proof. Must show that $\{x : x \prec y\} = \text{conv}(P_1 y, \ldots, P_n y)$ where P_j is a permutation matrix for $j = 1, \ldots, n$. We start by showing that $\{x : x \prec y\} \subseteq \text{conv}(P_1 y, \ldots, P_n y)$. For any $x \in \{x : x \prec y\}$ we have

$$\sum_{i=1}^{m} x_i = \sum_{j=1}^{m} y_i$$

by definition. Since $y \prec Py$ for any permutation matrix we have

$$\sum_{i=1}^{m} x_i = \sum_{j=1}^{m} y_i = \sum_{i=1}^{m} Py_i \in \sum_{j=1}^{n} \lambda_j P_j \sum_{i=1}^{m} y_i$$

where $\sum_{j=1}^{n} \lambda_j = 1$. Since the convex hull of a set is the set of all convex combinations of the elements in the set, we have

$\{x : x \prec y\} \subseteq \text{conv}(P_1 y, \ldots, P_n y)$.

Now, we will show the converse, $\text{conv}(P_1 y, \ldots, P_n y) \subseteq \{x : x \prec y\}$. $\text{conv}(P_1 y, \ldots, P_n y)$ consists of elements on the form $x = \sum_{j=1}^{n} \lambda_j P_j y$, where $\sum_{j=1}^{n} \lambda_j = 1$. We must show that $x \prec y$. We let y^* be the vector where the components of y are arranged such that $y_1^* \geq y_2^* \geq \ldots \geq y_n^*$ and let $y^j = P_j y$. Because $\sum_{i=1}^{k} y_i^j \leq \sum_{i=1}^{k} y_i^*$ for $k < n$, we can write

$$\sum_{i=1}^{k} x_i = \sum_{i=1}^{k} \sum_{j=1}^{t} \lambda_j P_j y_i = \sum_{i=1}^{k} \sum_{j=1}^{t} \lambda_j y_i^j \leq \sum_{i=1}^{k} y_i^* \sum_{j=1}^{t} \lambda_j = \sum_{i=1}^{k} y_i^*$$

for $k < n$ and $\sum_{j=1}^{t} \lambda_j = 1$,

Also, because $\sum_{i=1}^{n} y_i^j = \sum_{i=1}^{n} y_i^*$, we have

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} \sum_{j=1}^{t} \lambda_j P_j y_i = \sum_{i=1}^{n} \sum_{j=1}^{t} \lambda_j y_i^j = \sum_{i=1}^{n} \sum_{j=1}^{t} \lambda_j y_i^* = \sum_{i=1}^{n} y_i^* \sum_{j=1}^{t} \lambda_j = \sum_{i=1}^{n} y_i^*$$

where $\sum_{j=1}^{t} \lambda_j = 1$.

Hence, $x \prec y$, and therefore $\text{conv}(P_1 y, \ldots, P_n y) = \{x : x \prec y\}$. □

Next property, which considers the relationship between the vectors x and y, concerns linear transformations called T-transformations, T, which are defined to be transformations on the form

$$T = \lambda I + (1 - \lambda) Q$$

where $0 \leq \lambda \leq 1$ and Q is a permutation matrix that interchanges two entries. Then, we have the following lemma.

Lemma 7. *If $x \prec y$, then x can be derived from y by successive applications of a finite number of T-transformations.*

Proof. T applied once on $y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$, where $y_1 \geq \ldots \geq y_n$, gives

$$Ty = \lambda Iy + (1-\lambda)Q = \begin{pmatrix} \lambda y_1 \\ \vdots \\ \lambda y_{j-1} \\ \lambda y_j \\ \lambda y_{j+1} \\ \vdots \\ \lambda y_{k-1} \\ \lambda y_k \\ \lambda y_{k+1} \\ \vdots \\ \lambda y_n \end{pmatrix} + \begin{pmatrix} y_1 \\ \vdots \\ y_{j-1} \\ y_k \\ y_{j+1} \\ \vdots \\ y_{k-1} \\ y_j \\ y_{k+1} \\ \vdots \\ y_n \end{pmatrix} - \begin{pmatrix} \lambda y_1 \\ \vdots \\ \lambda y_{j-1} \\ \lambda y_k \\ \lambda y_{j+1} \\ \vdots \\ \lambda y_{k-1} \\ \lambda y_j \\ \lambda y_{k+1} \\ \vdots \\ \lambda y_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{j-1} \\ \lambda y_j + (1-\lambda)y_k \\ y_{j+1} \\ \vdots \\ y_{k-1} \\ \lambda y_k + (1-\lambda)y_j \\ y_{k+1} \\ \vdots \\ y_n \end{pmatrix}$$

We call this vector x. Each time T is applied on y, this process will be repeated and we get a similar vector, only with an increasing number of entries on the form $\lambda y_j + (1-\lambda)y_k$ for different λ's. For simplicity, we will only show that $x \prec y$ for the vector above. For multiple applications of T on y, the proof will be similar. We start by showing that $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$

$$\begin{aligned} \sum_{i=1}^{n} x_i &= y_1 + \ldots + \lambda y_j + (1-\lambda)y_k + \ldots + y_k + (1-\lambda)y_j + \ldots y_n \\ &= y_1 + \ldots + y_j + \ldots + y_k + \ldots + y_n \\ &= \sum_{i=1}^{n} y_i \end{aligned}$$

Secondly, we must show that $\sum_{i=1}^{t} x_i \leq \sum_{i=1}^{t} y_i$ for any $t < n$. We have 3 cases:

Case 1: $t \leq j$ **:** We get

$$\sum_{i=1}^{t} x_i = y_1 + \ldots + y_t = \sum_{i=1}^{t} y_i$$

Hence, the statement holds in fact by equality.
Case 2: $j < t < k$**:** This gives

$$\sum_{i=1}^{t} x_i = y_1 + \ldots + \lambda y_j + (1-\lambda)y_k + \ldots + y_t = y_1 + \ldots + \lambda(y_j - y_k) + y_k + \ldots + y_t$$

Since $\lambda \leq 1$ and $y_j - y_k \geq 0$, we have

$$\lambda(y_j - y_k) + y_k \leq y_j - y_k + y_k = y_j$$

So we get

$$\sum_{i=1}^{t} x_i = y_1 + \ldots + \lambda(y_j - y_k) + y_k + \ldots + y_t \leq y_1 + \ldots + y_j + \ldots + y_t = \sum_{i=1}^{t} y_i$$

and the inequality holds.
Case 3: $t \geq k$**:** Then we have

$$\begin{aligned}\sum_{i=1}^{t} x_i &= y_1 + \ldots + y_t \\ &= y_1 + \ldots + \lambda y_j + (1-\lambda)y_k + \ldots + y_k + (1-\lambda)y_j + \ldots y_t \\ &= y_1 + \ldots + y_j + \ldots + y_k + \ldots + y_t \\ &= \sum_{i=1}^{t} y_i\end{aligned}$$

And again we get that the statement holds by equality. □

Now, that majorization and some of its properties have been presented, we continue by relating this concept to doubly stochastic matrices by the following theorem.

Theorem 8. Hardy, Littlewood, Pólya (1929) *$x \prec y$ if and only if there exists a doubly stochastic matrix, A, such that $x = yA$*

The following proof is based on the proof given in the book by Marshall and Olkin [9].

Proof. $\Leftarrow$: By theorem 11 by Birkhoff and von Neumann presented later, all doubly stochastic matrices can be written as a convex combination of permutation matrices P_i, i.e. $A = \sum_i \lambda_i P_i$ where $\sum_i \lambda_i = 1$ and $A \in \Omega_n$. Then,

$$x = yA = y\sum_i \lambda_i P_i = \sum_i \lambda_i (yP_i)$$

which is a convex combination of permutations of y, i.e. $x \in \{x : x \prec y\}$ which is equal to $\text{conv}(P_1 y, \ldots, P_k y)$ by corollary 6. Hence, $x \prec y$.

$\Rightarrow$: Assume $x \prec y$. The T-transformation defined above is doubly stochastic by definition. By lemma 7, we can write

$$x = T^k y$$

for some k, which corresponds to multiple multiplications by some doubly stochastic matrix. This product will be a doubly stochastic matrix since the product of doubly stochastic matrices also is doubly stochastic. Hence, $x = Ay$ where A is some doubly stochastic matrix. □

Chapter 4

The Assignment Problem

By limiting the LP problems defined in (2.2) such that $A \in P_n$, we get what we call the assignment problem. In this chapter we will define and illustrate such problems by presenting some algorithms, theoretical results and applications. Most of the theory presented in this chapter is based on the article by Burkhard an Çela [3].

4.1 Introducing the problem

In the assignment problem, we want to assign n men/factories/workers etc. to n women/buyers/jobs etc. in a best possible way. We can think of this as finding a bijective mapping, ϕ, of a finite set, $N = \{1, 2, ...n\}$, into itself and will correspond to an $n \times n$ matrix, $X = (x_{ij})$, where

$$\begin{aligned} x_{ij} &= 1 \quad \text{for} \quad j = \phi(i) \\ x_{ij} &= 0 \quad \text{for} \quad j \neq \phi(i) \end{aligned}$$

X will have only one nonzero element in each row and each column, which will be one, hence $X \in P_n$. This give rise to what we call the *linear sum assignment problem* (LSAP).

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n} c_{ij} x_{ij} \\ \text{Subject to} \quad & \sum_{i=1}^{n} x_{ij} = 1 \quad \forall\, j = 1, ..., n \\ & \sum_{j=1}^{n} x_{ij} = 1 \quad \forall\, i = 1, ..., n \\ & x_{ij} \in \{0, 1\} \quad \forall\, i, j = 1, ...n \end{aligned} \tag{4.1}$$

with corresponding dual problem

$$\begin{array}{lll} \text{Maximize} & \sum_{i=1}^{n} u_i + \sum_{i=1}^{n} v_j & \\ \text{Subject to} & \sum_{i=1}^{n} u_i + v_j \leq c_{ij} & \forall\, i,j = 1,...,n \\ & u_i, v_j \in \mathbb{R} & \forall\, i,j = 1,...n \end{array} \tag{4.2}$$

4.2 Matchings

Assignment problems can also be expressed in graph notation. In order to do this, we must first transfer the concept of doubly stochastic matrices to a graph approach. For a problem on the form (4.1), where we let $x_{ij} \geq 0$ instead of $x_{ij} \in \{0,1\}$, we consider a bipartite graph $G = (V, W; E)$ where the vertex sets V and W represent the rows and columns of X respectively. The solution set of such problems will correspond to the Birkhoff polytope, Ω_n, which we can write in graph notation as

$$\Omega_n = \{x : \sum_{e \in \delta(v)} x(e) = 1 \text{ for } v \in V \cup W; x_e \geq 0 \;\forall\, e \in E\} \tag{4.3}$$

where the function $x : E \to \mathbb{R}$ is as defined in chapter 2.

Returning to the assignment problem in graph notation, the mapping ϕ is defined such that for $i \in V$ and $j \in W$, $\phi(i) = j$ if there exists an edge $e = (i,j) \in E$ with $x(e)$=1. This corresponds to $x_{ij} = 1$ for the matrix notation approach of this mapping. Since

$$\sum_{e \in \delta(v)} x(e) = 1 \;\forall\, v \in V \cup W$$

the vertices $v \in V$ and $w \in W$ can only be connected by the edge (v,w) whenever $\phi(v) = w$. Such edges constitute what we call a matching, which is defined as $M \subseteq E$ for a bipartite graph G where each matched vertex of G is associated with at most one edge in M. In some problems, we want to find a matching with as many edges as possible. These problems are called maximum matching problems. When every vertex in G is connected to exactly one edge in M, we say that we have a perfect matching. Hence, a perfect matching corresponds to an assignment which means that we can write the assignment problem as

$$\min\{\sum_{(v,w) \in M} c(v,w) : M \text{ is a perfect matching}\} \tag{4.4}$$

where $c(v, w)$ represents the cost related to the edge $e = (v, w)$ for $v \in V$ and $w \in W$.

The marriage theorem by Hall is closely related to this topic and concerns the existence perfect matchings in bipartite graphs. In the theorem we let V be a group of n men, W be a group of n women, and the edges in E are connections between them, i.e. possible marriages. If we let each man make a list of the women he is interested in marrying, marriages between each of the n men and the women on their respective lists will only be possible if for any integer, $k \in [1, n]$, the union of any k of the lists contains at least k women. Or mathematically speaking,

Theorem 9. (Marriage theorem, Hall, 1935) *Let $G = (V, W; E)$ be a bipartite graph. G contains a perfect matching (marriage) if and only if $|V| = |W|$ and for all subsets V^* of V*

$$|V^*| \leq |N(V^*)|$$

In the theorem, $N(v)$ denotes the set of neighbours of a vertex $v \in V$, i.e. the set of vertices $w \in W$ that are connected to v by an edge $e \in E$. This means that $N(V^*) = \bigcup_{v \in V^*} N(v)$, i.e. the subset $W^* \subseteq W$ of vertices $w \in W$ that are connected to vertices in V^* by edges in E. As the theorem states, a perfect matching is only possible if this set is bigger than the set $\bigcup_{v \in V^*} v$ for any set V^*.

Proof. $\Rightarrow$: Suppose G contains a perfect matching M. Then, for each $v \in V^*$ there exists a corresponding edge $e_{vw} \in M$ for $w \in W$, resulting in $|N(V^*)| \geq |V^*|$.

$\Leftarrow$: Suppose $|V^*| \leq |N(V^*)| \ \forall \ V^* \subseteq V$. We start by choosing a $v \in V$ randomly and then let $V^* = \{v\}$. Since $|V^*| = 1$, we must have $|N(V^*)| \geq 1$. Hence, there exists at least one edge from v to w for $w \in W$. This shows that for any $v \in V$ there is at least one edge $e_{vw} \in E$ incident to v. Now, we construct a marriage M by adding edges e_{vw} for each $v \in V$. If we get that to vertices $u, v \in V$ is connected to the same $w \in W$, i.e. we have e_{uw} and e_{vw} for $u \neq v$ for some $w \in W$, we replace the vertex w in the edge e_{uw} by another vertex adjacent to u. This is possible since for the set $\{u, v\}$, we have $|\{u, v\}| \leq |N(\{u, v\})|$. When we have done this for all vertices $v \in V$ we have a matching M for which $|M| = |V|$, i.e. M is a perfect matching. $\square$

4.3 König's Theorem

For an assignment problem with a corresponding $n \times n$ matrix, X, we know that the rest of the entries in the ith row and the jth column must be zero once we have assigned element i to element j because then, $x_{ij} = 1$. Hence, rows and columns corresponding to matched elements can only have one nonzero element, which means that if the graph represents a perfect matching, the corresponding matrix can only have n nonzero elements in total. We let $|M|$ denote the number of matched edges we have in a given bipartite graph. We now know that $|M| \leq n$.

A covering is defined to be the set of straight lines necessary in order to cover all such nonzero elements, i.e. the entries equal one. Since we only have one nonzero element in each row and each column, we need at least the same number of straight lines as the number of nonzero elements. In fact, these numbers must be equal, which is stated in the following theorem.

Theorem 10. *(*König's Theorem) *Let G be a bipartite graph. The size of a maximum matching of G is equal to the size of a minimum covering of G.*

$$max\ |M| = min\ |C|$$

Proof. The fact that max $|M| \leq$ min $|C|$ is trivial and was explained above.

It remains to show that max $|M| \geq$ min $|C|$. For a permutation matrix, we let the elements equal one be covered by rows L_1 and columns L_2, such that $|L_1| + |L_2| = |C|$, with as few lines in total as possible, i.e. C is a minimum cover. Then, we define a function f_1, where $f_1(L_1)$ is the set of columns necessary in order to cover the elements equal one covered by the rows in L_1. Obviously, $|L_1| \leq |f_1(L_1)|$. Otherwise, the elements covered by L_1 could be covered by less columns, which contradicts the fact that C is a minimum cover. Also, for any subset $S_1 \subseteq L_1$ we have $|f(S_1)| \geq |S_1|$ by the same argument. Then by the Marriage theorem the elements in S_1 must be matched. Hence, we have $|S_1|$ matched elements. By letting $S_1 = L_1$, we get $|L_1|$ matched elements.

Similarly, by defining f_2 to be a function where $f_2(L_2)$ is the set of rows we need in order to cover all elements equal one covered by the columns in L_2, we get $|L_2|$ new matched elements by the same arguments. Hence, we have $|L_1| + |L_2| = |C|$ matched elements in total, which means that min $|C| = |M| \leq$ max $|M|$.

This finishes the proof, and we have now proved that max $|M| =$ min $|C|$ is in fact true. □

Example 5. *We consider the identity matrix for* $n = 3$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

We observe that it has 3 elements equal 1, which corresponds to the fact that we can have no more than 3 matched elements, i.e. $max|M| = 3$. *We want to cover these entries with as few straight lines as possible. There exist many ways in doing so. E.g.*

$$\begin{pmatrix} \cancel{1} & 0 & 0 \\ 0 & \cancel{1} & 0 \\ 0 & 0 & \cancel{1} \end{pmatrix}, \begin{pmatrix} \cancel{1} & \cancel{0} & \cancel{0} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

all using 3 lines in order to cover the entries equal 1, i.e. $min|C| = 3$. *Hence,* $max|M| = 3 = min|C|$, *which is what we expected according to König's theorem.*

Next, we consider

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

where we observe that we can have at most 2 matched elements, i.e. $max|M| = 2$. *To cover the entries equal 1 with as few straight lines as possible, we could for instance use these lines*

$$\begin{pmatrix} \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0} \\ 0.5 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We observe that 2 lines are necessary, i.e. $min|C| = 2$. *Hence,* $max|M| = 2 = min|C|$, *which again is consistent with König's theorem.*

4.4 Algorithms

In order to solve the LSAP problems (4.1) that we now have presented, there exist many algorithms. In this section we will present three such algorithms

which all work with a pair of an infeasible primal solution, x_{ij}, of (4.1) and a feasible dual solution, u_i, v_j, of (4.2). Therefore, these algorithms are called primal-dual algorithms. This pair fulfills the complementarity slackness conditions, $x_{ij}(c_{ij} - u_i - v_j) = 0$ for $1 \leq i, j \leq n$, which are kept fulfilled through the iterations necessary in order to reach an optimal feasible primal solution.

4.4.1 The Hungarian Method

The first algorithm to be presented is the Hungarian method, which was developed by Harold Kuhn in 1955 inspired by the Hungarian mathematicians Dénes König and Jenö Egervary, thereby the name. In this method we start with an infeasible primal solution, x_{ij}, and obtain a feasible dual solution, u_i, v_j, by performing row and column reductions on a given cost matrix, C.

In the row reduction, we find the smallest element of each row and subtract it from all elements of that row

$$u_i = \{\min c_{ij} \text{ for } 1 \leq j \leq n\} \text{ for } 1 \leq i \leq n$$

Then, we obtain a new cost matrix, C^*, with elements

$$c^*_{ij} = \{c_{ij} - u_i \text{ for } 1 \leq j \leq n\} \text{ for } 1 \leq i \leq n$$

on which we perform the column reduction by identifying the smallest element of each column

$$v_j = \min \{c^*_{ij} \, 1 \leq i \leq n\} \text{ for } 1 \leq j \leq n$$

and subtract it form all elements of that column, which results in the final cost matrix C^{**} with elements

$$c^{**}_{ij} = \{c^*_{ij} - v^*_j \text{ for } 1 \leq i \leq n\} \text{ for } 1 \leq j \leq n$$

The relationship between the original and the final cost matrix will be $c^{**}_{ij} = c_{ij} - u_i - v_j \geq 0$. This process will not change the problem because reducing the value of all the entries of a row (or column) of a matrix by the same number, will not change the optimality of the solution. E.g. if we are assigning n jobs to n workers, and each worker is lowering his or her price of the same job by the same value, the solution will remain the same because each of the workers must be assigned to one of the jobs. As a result, the optimal value will naturally be different.

The row and column reduced cost matrix C^{**} has now at least one zero

in each row and each column. The positions of these zeros represent the possible assignments we can make. I.e. if $c_{ij}^{**} = 0$ the ith worker can possibly be assigned to the jth job. When the assignments are made, the primal solution is updated such that we get

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is in the assignment} \\ 0 & \text{otherwise} \end{cases}$$

Since we for (i,j) in the assignment have $c_{ij} - u_i - v_j = 0$ and $x_{ij} = 1$, and $x_{ij} = 0$ otherwise, the complementarity slackness conditions,
$x_{ij}(c_{ij} - u_i - v_j) = 0$, are satisfied. We also need our primal solution, X, to be feasible, which means that we need n assigned elements, one for each row and one for each column. Since C^{**} might have more than n zeros, which assignments to make is not necessarily trivial. Therefore, the following procedure will help us determine this.

1. For each row, if there is only one zero, add it to the assignment. If there is more, wait.
2. When (i,j) is assigned, the other zeros in row i and column j are no longer assignable.
3. If there are still assignable zeros, we continue with the columns in the same manner.
4. This process is repeated until there are no more assignable zeros.

If we now have an assignment containing n elements, the solution is optimal and we are done. If not, we need to update the dual variables by the following procedure on the cost matrix C^{**}

5. Mark all unassigned rows.
6. If a marked row has unassigned zeros, mark corresponding columns.
7. If a marked column has an assigned zero, mark corresponding row.
8. Repeat step 6-7 for all marked rows.
9. Draw lines through the unmarked rows and the marked columns.

We observe that the number of lines we get represents the number of assigned elements, which is consistent with König's theorem. The procedure continues by updating the elements of the cost matrix C^{**}

10. Let δ be the smallest entry with no line through it.
11. Subtract δ from all entries having no line trough them.
12. Add δ to all entries having two lines through them.
13. Let the rest of the entries remain the same.

If we let I be all marked rows and J be all unmarked columns, the step 10-13 can be presented as

$$c_{ij}^{***} = \begin{cases} c_{ij}^{**} - \delta & \text{if } i \in I,\ j \in J \\ c_{ij}^{**} + \delta & \text{if } i \notin I,\ j \notin J \\ c_{ij}^{**} & \text{otherwise} \end{cases}$$

Where c_{ij}^{***} is the elements of our new cost matrix. Then, we start over at (1) with C^{***} as our cost matrix and repeat the process until we have n assigned elements, i.e. a primal feasible optimal solution.

Example 6. *We have 4 workers and 4 jobs to be done, i.e. a* 4×4 *problem, with a given cost matrix*

$$C = \begin{pmatrix} 90 & 75 & 75 & 80 \\ 35 & 85 & 55 & 65 \\ 125 & 95 & 90 & 105 \\ 45 & 110 & 95 & 115 \end{pmatrix}$$

where c_{ij} *denotes the price of the ith job by the jth worker. We use the Hungarian method to solve this problem and start by determining* u*, the vector containing the smallest elements of each row, which we use in order to compute* C^*

$$u = \begin{pmatrix} 75 \\ 35 \\ 90 \\ 45 \end{pmatrix} \quad \Rightarrow \quad C^* = \begin{pmatrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{pmatrix}$$

Then, we repeat this for the columns and obtain v *and* C^{**}

$$v = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 5 \end{pmatrix} \quad \Rightarrow \quad C^{**} = \begin{pmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{pmatrix}$$

Now, we can start the assignment process. In the first row we have 3 zeros, so we do nothing. In the next row there is only one zero, and therefore we add it to our solution. The same goes for the third row. For the last row, the only zero is not assignable because an element in the corresponding column has already been assigned. We then proceed to the columns. The first column already has an assigned element. In the second column there is only one zero, which therefore is added to our solution. Now, we are done because all other zeros are no longer assignable and we get

$$\begin{pmatrix} 15 & \boxed{0} & 0 & 0 \\ \boxed{0} & 50 & 20 & 25 \\ 35 & 5 & \boxed{0} & 10 \\ 0 & 65 & 55 & 65 \end{pmatrix}$$

where the entries corresponding to assigned elements are marked by a square. We continue the algorithm by marking rows without assignments, the corresponding column(s) containing a zero, and the corresponding rows of possible assignments for these columns. Then, we draw lines through all unmarked rows and all marked columns until we get

$$\begin{array}{c} \\ \\ \checkmark \\ \checkmark \\ \checkmark \end{array} \begin{array}{c} \\ \\ \begin{pmatrix} \cancel{15} & \boxed{0} & 0 & \cancel{0} \\ \boxed{0} & 50 & 20 & 25 \\ \cancel{35} & 5 & \boxed{0} & \cancel{10} \\ 0 & 65 & 50 & 65 \end{pmatrix} \end{array}$$

We identify the smallest element without any lines, $\delta = 20$, and add this number to all entries having two lines, subtract it from all entries having no lines and let the rest of the entries (those having one line) remain the same. This results in

$$\begin{pmatrix} 35 & 0 & 0 & 0 \\ 0 & 30 & 0 & 5 \\ 55 & 5 & 0 & 10 \\ 0 & 45 & 30 & 45 \end{pmatrix}$$

By repeating the assigning procedure we get

$$\begin{pmatrix} 35 & \boxed{0} & 0 & 0 \\ 0 & 30 & 0 & 5 \\ 55 & 5 & \boxed{0} & 0 \\ \boxed{0} & 45 & 30 & 45 \end{pmatrix}$$

where we again mark rows, columns and draw lines according to the algorithm and get

$$\begin{array}{c} \quad\checkmark \quad\quad\quad \checkmark \\ \begin{array}{c}\\ \checkmark \\ \checkmark \\ \checkmark \end{array} \begin{pmatrix} \cancel{35} & \boxed{0} & \cancel{0} & \cancel{0} \\ 0 & 30 & 0 & 5 \\ 55 & 5 & \boxed{0} & 10 \\ \boxed{0} & 45 & 30 & 45 \end{pmatrix} \end{array}$$

Now, $\delta = 5$*, which results in*

$$\begin{pmatrix} 40 & 0 & 5 & 0 \\ 0 & 25 & 0 & 0 \\ 55 & 0 & 0 & 5 \\ 0 & 40 & 30 & 40 \end{pmatrix}$$

Again, we repeat the assigning process, which finally gives us

$$\begin{pmatrix} 40 & \boxed{0} & 5 & 0 \\ 0 & 25 & 0 & \boxed{0} \\ 55 & 0 & \boxed{0} & 5 \\ \boxed{0} & 40 & 30 & 40 \end{pmatrix}$$

where we observe that we have 4 assigned elements, one for each row and each column. This means that we have obtained a primal feasible optimal solution, and therefore, we are done. So, the first job goes to the second worker, the second job goes to the third worker, the third job goes to the forth worker and the forth job goes to the first worker, which gives a total cost of $75 + 65 + 90 + 45 = 275$*, which is the minimum total cost.*

4.4.2 The Shortest Augmenting Path Algorithms

Whereas the Hungarian method has a matrix approach to the problem (4.1), we will now present an algorithm using graph notation. In Shortest Augmenting Path Algorithms, we let $M \subseteq E$ be some matching for a bipartite graph, $G = (V, W; E)$, and define an augmenting path to be a path connecting two vertices not connected to an edge in M, where the edges are alternately in and out of the matching. Such algorithms are based on the fact that this path must necessarily have one more edge in $E \backslash M$ than in M, and therefore, by swapping these sets, the matching will increase by one.

As in the Hungarian method, we start by an infeasible primal solution x_{ij} for $1 \leq i, j \leq n$, and a feasible dual solution u_i, v_j for $1 \leq i, j \leq n$, which fulfill

the complementarity slackness conditions. We construct a bipartite graph, $\tilde{G} = (V, W; \tilde{E})$, where

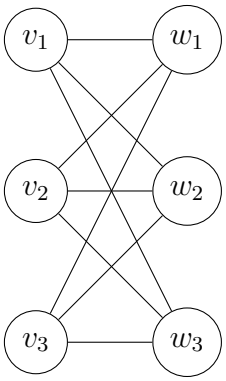
$$\tilde{E} = \{(i,j) : (i,j) \in E \backslash M\} \cup \{(j,i) : (i,j) \in M\}$$

which means that $\tilde{G}$ will be a directed graph containing some edges from V to W, which we will say are pointing forwards, and some edges pointing backwards, i.e. from W to V. This makes finding an augmenting path possible. We add the reduced costs, $\bar{c_{ij}} = c_{ij} - u_i - v_j$, to the corresponding forwards edges, whereas the costs of the backwards edges are set to zero. Hence, we have a weighted, directed bipartite graph, $\tilde{G}$, for which we want to find an augmenting path (if it exists).

The algorithm starts with an empty matching. We select a free vertex in V and compute the augmenting paths from this vertex to all vertices in W, which for this first iteration only will consist of one edge, which will be unmatched. Then, we select the shortest of these paths, i.e. the path with the lowest corresponding cost. This edge is then included in the matching, i.e. we now have $|M| = 1$. In general, we compute all augmenting paths from a selected vertex in V to W and select the shortest one. This path will begin and end with edges in $E \backslash M$, i.e. unmatched edges. Therefore, swapping the matched and the unmatched edges in the path results in the matching increasing by one for each iteration. Since an optimal solution corresponds to a perfect matching, i.e. $|M| = n$, we will obtain an optimal solution after n iterations.

Example 7. *Let V be the set of 3 men and W be the set of 3 women, which we want to match into pairs. Each man can be paired with any woman. Hence, we get the following bipartite graph*

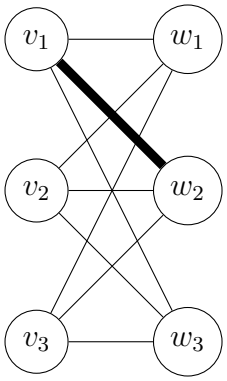
$$\begin{aligned} G &= (V, W; E) \\ &= (\{v_1, v_2, v_3\}, \{w_1, w_2.w_3\}| \\ &\{(v_1, w_2), (v_1, w_2), (v_1, w_3), (v_2, w_1), (v_2, w_2), (v_2, w_3), (v_3, w_1), (v_3, w_2), (v_3, w_3)\}) \end{aligned}$$



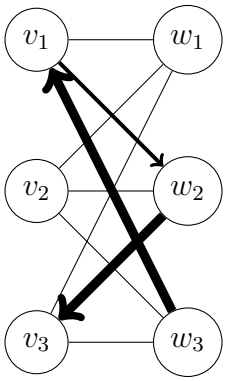
We start by having no couples, $M = \emptyset$, and all edges directed forwards. We assume all combinations of men and women are not equally ideal, and we therefore associate each possible combination with a certain level of misery, which will be presented as a cost matrix C, where the elements c_{ij} denotes the misery related to matching the ith man and the jth woman. For this example we let

$$C = \begin{pmatrix} 5 & 2 & 3 \\ 2 & 4 & 1 \\ 7 & 5 & 9 \end{pmatrix}$$

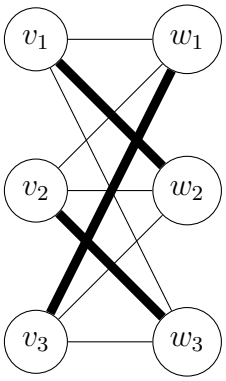
We start the algorithm by randomly selecting the vertex v_1. Then, we identify the edge with the smallest associated cost, connected to this vertex, i.e. the shortest path from v_1 to W. We observe that this will be the path (v_1, w_2). By swapping the matched and the unmatched edges in this path, $(v_1, w_2) \in M$ and $|M| = 1$, which corresponds to the following graph



where the thick line represents a matched edge. We repeat this procedure and get a new path consisting of the 3 edges $(v_3, w_2), (w_2, v_1), (v_1, w_2)$. Again, we swap matched and unmatched edges, which results in a matching of two edges and the graph



When we repeat this procedure one more time, we get a path consisting of the 5 edges $(v_2, w_3), (w_3, v_1), (v_1, w_2), (w_2, v_3), (v_3, w_1)$*, and by swapping the matched and unmatched edges of this path, we get a matching of 3 edges. Hence, we are done since this corresponds to a perfect matching and will therefore be an optimal solution*



This solution corresponds to a total misery of $1 + 2 + 7 = 10$*.*

4.4.3 The Auction Algorithm

The last algorithm to be presented is the auction algorithm, which has a matrix approach to the LSAP problem, where we let the rows represent the customers and the columns represent the items to be sold. Unlike the previously presented algorithms, this algorithm was originally developed to solve maximization problems, which in an auction setting means that we want to make the customers buy the items for the highest total price possible when some given constraints must be satisfied. It has also been developed a minimization approach of this algorithm, but in this thesis we will present the original version. Therefore, we want to solve the following problem

$$\begin{aligned}
\text{Maximize} \quad & \sum_{i=1}^{n} c_{ij} x_{ij} \\
\text{Subject to} \quad & \sum_{i=1}^{n} x_{ij} = 1 \quad \forall\, j = 1, ..., n \\
& \sum_{j=1}^{n} x_{ij} = 1 \quad \forall\, i = 1, ..., n \\
& x_{ij} \in \{0, 1\} \quad \forall\, i, j = 1, ...n
\end{aligned}$$

where the c_{ij} refers to how much customer i is willing to pay for item j. Also, a set $A(i)$ is given, denoting the items customer i is interested in buying. Hence, $A(i)$ are the items that customer i can be assigned to. As before, the algorithm starts with a pair of an infeasible primal solution, x_{ij}, and a

feasible dual solution, u_i, v_j, fulfilling the complementarity slackness conditions, $x_{ij}(c_{ij} - u_i - v_j) = 0$. v_j denotes the price of item j, i.e. the price a person assigned to this item must pay, and $u_i = \max_{j \in A(i)}\{c_{ij} - v_j\}$ is the profit margin of customer i corresponding to item j. The primal solution will be feasible when all objects are assigned to different customers and optimal when the total sale price, $c_{ij}x_{ij}$, is as high as possible. The pair of the primal and dual solutions are updated iteratively by a process consisting of two phases, the bidding phase and the assignment phase, which preserves the complementary slackness conditions throughout its execution.

Bidding phase: For each free customer, i, we compute the "current value", p_{ij}, of each item $j \in A(i)$ by

$$p_{ij} = c_{ij} - v_j$$

and find the items, j^* and j^{**}, corresponding to the highest and second highest such values respectively

$$\begin{aligned} p_{ij^*} &= \max_{j \in A(i)} p_{ij} \\ p_{ij^{**}} &= \max_{j \in A(i), j \neq j^*} p_{ij} \end{aligned}$$

We then say that customer i bids on item j^*, and that item j^* receives a bid from customer i of the value

$$b_{ij^*} = v_{j^*} + p_{ij^*} - p_{ij^{**}} + \epsilon = c_{ij^*} - p_{ij^{**}} + \epsilon$$

As long as $v_{j^*} + \epsilon < b_{ij^*} < v_{j^*} + p_{ij^*} - p_{ij^{**}} + \epsilon$, the algorithm will work

Assignment phase: For each item, j, we let $S(j)$ be the set of customers from which item j received a bid in the bidding phase. If $S(j) \neq \emptyset$, we increase v_j to the highest bid, i.e. set $v_j = \max_{i \in S(j)} b_{ij}$. We let i^* denote the customer in the set $S(j)$ having the highest bid. If there existed any pair (i, j) where $i \neq i^*$ in the assignment, we remove this and add (i^*, j) instead.

These two phases will happen parallelly. For each iteration, the assignment matrix, X, will change, but this will not necessarily result in more assigned elements. Hence, items can change buyer, but once it has been assigned, it remains assigned throughout the remainder of the algorithm. Also, there will always exist items that never have been assigned, except at termination.

4.5 Applications

Now, that we have presented the theoretical background of assignment problems and some algorithms that will solve them, we will continue by applying

these algorithms to some real world problems that can be presented as assignment problems.

4.5.1 An Elevator system

A skyscraper consists of many floors, and consequently the people in the building will often need to travel distances of many floors numerous times a day in order to reach their desired destinations. For this to happen efficiently, a functioning elevator system is necessary. We let k be the number of floors in our building and n be the number of elevators. Naturally, $k \gg n$, but the relationship between k and n must be reasonable. We also let m denote the number of passengers or groups of passengers the elevators are operating at a time. Since m cannot exceed n, a queue will be started where the $n+1$th person to hit the press button will be number one. Our job is to find the most efficient way of transporting these m passengers to their desired destination floors by our n elevators, i.e. the shortest total travel distance for the elevators combined. Since the distances our passengers want to travel is already given, we will not be able to minimize this distance. However, the distance an elevator has to travel in order to reach a passenger from its current position can be minimized. Hence, this can be described as a shortest path problem where we want to minimize the total distance the elevators are traveling without any passengers. If we let h_j be the position (floor number) of elevator j and p_i be the position of passenger i, the distance elevator j has to travel in order to reach passenger i will be $|p_i - h_j|$. We call this distance c_{ij}, which gives rise to an $n \times n$ cost matrix C. For situations when $m < n$, i.e.passenger $t \leq n$ is nonexistent, we let $c_{tj} = 0 \ \forall \ j$. By this we get a minimization problem, which will be on the form (4.1), where $x_{ij} = 1$ whenever elevator j will be sent off to transport passenger i .

4.5.2 Mathematical Holmenkollen relay race

The annual Holmenkollen relay race, which has its starting and ending point at Bislett in Oslo, has 15 legs. Therefore, our participating team consists of 15 runners. They are all both dedicated and determined to do this properly and have therefore already practiced and done some test runs where each member of the team has been running all of the legs of the relay. The best running time for each team member for each leg has been recorded and put into a 15×15 matrix, C, which will be our cost matrix. We let c_{ij} denote the ith runners best time on leg j. In order to get the best result in the

actual race, we need to assign our runners to the legs of the relay such that the total running time is minimized. We define the elements of our solution matrix, X, such that $x_{ij} = 1$ if the ith runner is assigned to the jth leg, and $x_{ij} = 0$ otherwise, which gives rise to a minimization problem on the form (4.1). We now present an illustrating example.

Example 8. *We let A, B, C, D, E, F, G, H, I, J, K, L, M, N and O denote the team members, which we want to assign to the 15 legs of the Holmenkollen relay race. They have all been training for some time and the test runs have resulted in the following data set*

$$\begin{pmatrix}
6.18 & 5.11 & 2.30 & 7.01 & 4.05 & 5.39 & 8.56 & 7.20 & 2.48 & 10.28 & 6.29 & 1.38 & 4.12 & 4.11 & 2.48 \\
5.51 & 4.14 & 2.22 & 6.46 & 5.14 & 8.19 & 9.01 & 6.31 & 2.21 & 12.31 & 5.59 & 1.14 & 4.22 & 3.19 & 2.51 \\
4.03 & 3.55 & 2.31 & 6.03 & 4.59 & 5.09 & 7.56 & 6.06 & 2.29 & 10.01 & 4.59 & 1.22 & 3.59 & 3.31 & 2.48 \\
3.56 & 4.22 & 2.45 & 6.45 & 4.01 & 5.03 & 7.45 & 6.31 & 2.12 & 10.12 & 5.12 & 1.43 & 3.57 & 4.01 & 2.26 \\
3.40 & 4.01 & 2.22 & 7.01 & 4.56 & 5.23 & 7.23 & 6.04 & 2.25 & 10.05 & 5.56 & 1.23 & 3.54 & 3.12 & 2.23 \\
3.56 & 5.33 & 2.56 & 6.45 & 4.02 & 5.34 & 7.46 & 6.13 & 2.39 & 9.56 & 5.35 & 1.34 & 4.05 & 4.03 & 2.18 \\
5.39 & 4.59 & 3.01 & 8.01 & 4.34 & 6.12 & 8.45 & 8.12 & 2.41 & 10.56 & 7.32 & 1.34 & 4.23 & 4.34 & 3.09 \\
3.59 & 3.56 & 2.24 & 6.07 & 4.19 & 5.35 & 7.34 & 6.12 & 2.23 & 10.01 & 5.14 & 1.32 & 4.14 & 3.12 & 2.45 \\
3.20 & 4.56 & 2.45 & 6.03 & 4.02 & 5.12 & 7.25 & 5.45 & 2.34 & 9.34 & 5.34 & 1.23 & 3.56 & 3.09 & 2.23 \\
3.45 & 4.34 & 2.34 & 6.45 & 4.36 & 5.01 & 7.36 & 5.26 & 2.21 & 8.45 & 5.50 & 1.20 & 3.56 & 3.45 & 2.31 \\
4.44 & 4.59 & 2.45 & 7.06 & 4.56 & 7.34 & 8.36 & 7.23 & 3.45 & 11.02 & 6.41 & 1.26 & 4.12 & 4.05 & 2.51 \\
4.03 & 4.17 & 2.19 & 6.05 & 4.13 & 4.49 & 7.36 & 5.59 & 2.24 & 8.56 & 5.34 & 1.32 & 3.48 & 2.59 & 2.43 \\
3.59 & 4.32 & 2.22 & 5.56 & 4.04 & 5.08 & 7.23 & 5.23 & 2.36 & 8.59 & 5.21 & 1.15 & 3.56 & 3.05 & 2.21 \\
5.29 & 5.09 & 2.47 & 6.41 & 4.58 & 5.34 & 8.12 & 6.22 & 2.22 & 10.01 & 5.59 & 1.12 & 3.50 & 4.01 & 2.45 \\
4.12 & 4.44 & 2.45 & 7.14 & 4.09 & 6.15 & 8.04 & 5.55 & 2.36 & 9.54 & 6.09 & 1.23 & 3.59 & 3.56 & 2.34
\end{pmatrix}$$

where the rows represent the team members in the order presented above and the legs 1-15 are represented by the columns. Since this is an LP problem, we can use the Simplex method in order to solve it. We observe that this is also an assignment problem for which we can use the solution algorithms presented earlier as well. In this example we will (i) solve this problem by using the Simplex method implemented in an OPL-CPLEX program and (ii) use one the algorithms just presented, more precisely the Hungarian method implemented in a Matlab program.

(i) ***Simplex Method***
We implement the problem as a OPL-CPLEX program in an minimization LP problem where we let the matrix above be the cost matrix, C, and X be the solution matrix, which eventually will be the 15×15 permutation matrix representing which assignments to make in order to achieve the best total running time possible. By running this program,

which is presented in the appendix, we get

$$X = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(ii) ***The Hungarian Algorithm***

Next, we use the Hungarian method by implementing the steps described in the algorithm in a Matlab program (see appendix A). By letting the program do these steps repeatedly until we have 15 assigned elements, we get an optimal solution, which is the same solution matrix, X, as we got by using the Simplex method. Hence, both methods for solving this problem return the best possible total running time for our team in the Holmenkollen relay race to be 61.38 *minutes, which is the optimal value with a corresponding order of the runners as*

I - H - B - M - A - D - E - O - G - J - C - K - N - L - F

Chapter 5

The Birkhoff and von Neumann Theorem

In this chapter we will present the main theorem of this thesis, namely the Birkhoff and von Neumann theorem. Most of this chapter will be spent on two different proofs of this theorem. The first proof is based on the proof presented in the book by Marshall and Olkin [9] and are using convexity theory with a matrix notation approach, whereas the second proof uses a graph approach of the theorem and is based on both graph theory and two theoretical results from LP theory, which will be presented and proved. But first of all, we will present the actual theorem and an illustrating example

Theorem 11. (Birkhoff (1946), von Neumann (1953)) *The permutation matrices constitute the extreme points of the set of doubly stochastic matrices. Moreover, the set of doubly stochastic matrices, Ω_n, is the convex hull of the permutation matrices.*

According to the definition of convex hull stated earlier, this means that the set of all doubly stochastic matrices, Ω_n, is the set of all convex combinations of the permutation matrices, P_n, which by this theorem will be the extreme points of Ω_n. Hence, if we let $A \in \Omega_n$, we can write

$$A = \sum_{j=1}^{n} \lambda_j P_j \tag{5.1}$$

where $P_1, ..., P_n$ are permutation matrices, $\lambda_j \geq 0$ for $j = 1, \ldots, n$ and $\sum_{j=1}^{n} \lambda_j = 1$. We illustrate this by the following example

Example 9. *We let*

$$A = \begin{pmatrix} 0.2 & 0.3 & 0.5 \\ 0.3 & 0.6 & 0.1 \\ 0.5 & 0.1 & 0.4 \end{pmatrix}$$

which will be an element of Ω_3*, and observe that we can write*

$$0.3\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 0.1\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} + 0.5\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + 0.1\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = A$$

where $0.3 + 0.1 + 0.5 + 0.1 = 1$*, i.e.* $\sum_{i=1}^{4} \lambda_i = 1$*. Hence, we have written* A *as a convex combination of 4 permutation matrices.*

Notice that we can also write

$$A = 0.2\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 0.4\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 0.1\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$
$$+0.2\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 0.1\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

where $0.2 + 0.4 + 0.1 + 0.2 + 0.1 = 1$*. Hence, the convex combination of* A *is not unique. However, there exist matrices that have an unique convex combination. For* $n \times n$ *problems, the Birkhoff polytope,* Ω_n*, will constitute all possible solution matrices, and because* $\Omega_n = conv(P_n)$*, the* $n!$ *vertices of* Ω_n*, which are equal to* P_n*, will determine* Ω_n*. For any selection of* $k+1$ *of these vertices, we get a k-simplex. By proposition 3, we know that every point in a simplex can be written as a unique convex combination of its vertices. Therefore, the convex combination of a matrix will be unique if the matrix is only contained in one of the simplices of the polytope in question. We observe that matrix* A *is at least contained in two simplices, the 3-simplex spanned by*

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and the 4-simplex spanned by

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix},$$

therefore, the convex combination cannot be unique. If we let a matrix approach any of the vertices of the polytope, the number of simplices that will

contain this matrix will decrease. Therefore, if a matrix is sufficiently close to a vertex, there will exist an unique convex combination of this matrix. If we let one or more of the elements of a matrix equal zero, we exclude some of the vertices from possibly being a part of any possible convex combination, and we therefore get fewer possible k-simplices that can contain this matrix. The matrix

$$B = \begin{pmatrix} 0.2 & 0.3 & 0.5 \\ 0.4 & 0.6 & 0 \\ 0.4 & 0.1 & 0.5 \end{pmatrix}$$

has a zero at $(2,3)$*, and hence, only the vertices*

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

can possibly be in its convex combination. Since the 3-simplex spanned by these 4 vertices is the only simplex that B *is contained in, the convex combiantion must be unique.*

5.1 Proof I

As mentioned initially, the first proof of the Birkhoff and von Neumann will have a matrix approach, which means that Ω_n and P_n will be considered matrices. Some properties from LP theory and convexity theory will be used.

Proof. We defined Ω_n to be a convex polytope in $\mathbb{R}^{n\times n}$ with matrices, A, satisfying

$$\begin{aligned} \sum_{j=1}^{n} a_{ij} &= 1 \quad \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} a_{ij} &= 1 \quad \text{for } j = 1, \ldots, n-1 \\ a_{ij} &\geq 0 \quad \text{for } i, j = 1, \ldots, n \end{aligned}$$

as its elements. We let the j-index in the second equality run up to $n-1$ only, because when $j = n$, the constraints are already given by the other equations. Hence, we have $2n-1$ equalities in total defining Ω_n.

Now, we assume $A \in \Omega_n \subset \mathbb{R}^{n\times n}$ is an extreme point of Ω_n. Therefore, we must show that A is a permutation matrix. We know that in order to determine an element of $\mathbb{R}^{n\times n}$ uniquely, at least n^2 linear equations are necessary. Since we maximally have $2n-1$ equations by the definition of Ω_n, at least $n^2 - (2n-1) = (n-1)^2$ of the entries of A must equal zero.

Therefore, at least one row must have $n-1$ zero-entries, which forces the nonzero entry of this row to be 1. The column corresponding to this entry must as a result have all other entries equal zero. Hence, this row and column satisfy the conditions for being part of a permutation matrix.

We now delete this row and column and obtain an $(n-1)\times(n-1)$ matrix A^*, which we must show is an extreme point of Ω_{n-1}. We assume for simplicity that we deleted the last row and column of A. For contradiction, we suppose that A^* is not an extreme point of Ω_{n-1}. Hence, we can write

$$A^* = \lambda A_1 + (1-\lambda)A_2$$

for some $A_1, A_2 \in \Omega_{n-1}$ and $0 \leq \lambda \leq 1$. Then,

$$A = \lambda \begin{pmatrix} A_1 & 0 \\ 0 & 1 \end{pmatrix} + (1-\lambda) \begin{pmatrix} A_2 & 0 \\ 0 & 1 \end{pmatrix}$$

where $\begin{pmatrix} A_1 & 0 \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} A_2 & 0 \\ 0 & 1 \end{pmatrix}$ are two elements of Ω_n. But since A is supposed to be an extreme point of Ω_n, we must have $A_1 = A_2$. Hence, A^* is an extreme point of Ω_{n-1}.

By using the same arguments as above, we state that A^* must have at least one row and one column with only one nonzero entry, which therefore must equal 1. Again, we delete this row and column from A^* and obtain an $(n-2)\times(n-2)$ matrix A^{**}, which is an extreme point of Ω_{n-2} by the same arguments as before. By repeating this process by induction on n, we eventually get $A^{(n-1)} = 1$, which is the permutation matrix of Ω_1. We have now showed that A consists of 0 and 1 elements only, and since $A \in \Omega_n$, A must be a permutation matrix.

Secondly, we must show that all $n \times n$ permutation matrices are extreme points of Ω_n. We let P be a permutation matrix and suppose for contradiction that P is not an extreme point of Ω_n, hence we can write

$$P = \lambda A_1 + (1-\lambda)A_2 \tag{5.2}$$

for some matrices $A_1, A_2 \in \Omega_n$ where $0 \leq \lambda \leq 1$. Because P is a permutation matrix, it can have entries equal 0 and 1 only. Therefore, (5.2) will only be satisfied if $A_1 = A_2$. Hence, P cannot be expressed as a midpoint of two other elements of Ω_n and must therefore be an extreme point of Ω_n, which means that the permutation matrices are the extreme points of Ω_n.

We conclude this proof be stating that since Ω_n is closed, it will be a polytope, and because a polytope is defined to be the convex hull of its extreme points, $\Omega_n = \text{conv}(P_n)$. □

5.2 Proof II

For the second proof of the Birkhoff and von Neumann theorem we will need two theoretical results, which we therefore will start by presenting.

Proposition 12. *Let A be an $n \times m$ matrix with rank m and consider the polyhedron $Q = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$. Let $x \in Q$. Then x is an extreme point of P if and only if x is a basic solution.*

Proof. $\Leftarrow$: Let x be a basic feasible solution of $Ax = b$, hence $x = A^{-1}b$. For contradiction we assume we can write $x = \frac{1}{2}x_1 + \frac{1}{2}x_2$ for some $x_1, x_2 \in P$, i.e. x is the midpoint of two points of Q. Since $x_1, x_2 \in Q$, we have

$$Ax_1 = b \Rightarrow x_1 = A^{-1}b$$

and similarly

$$Ax_2 = b \Rightarrow x_2 = A^{-1}b$$

Hence, $x_1 = A^{-1}b = x_2$, i.e. x cannot be written as a midpoint of two distinct points of Q and must therefore be an extreme point of Q.

$\Rightarrow$: Let $x \in Q$ be an extreme point of Q. Then x satisfies $Ax = b$. We must show that the columns of A are linearly independent. Assume there exists a λ such that $\sum_{j=1}^{n} \lambda_j a_j = 0$, where a_j represents the jth column of A. For linear independence, we must show that $\lambda_1 = \ldots = \lambda_n = 0$ is the only solution. Since $\sum_{j=1}^{n} \lambda_j a_j = 0$ supposedly, we can write

$$\begin{aligned}
\textstyle\sum_{j=1}^{n}(x_j a_j + \lambda_j a_j) = b \quad & \Rightarrow \textstyle\sum_{j=1}^{n} x_j a_j + \sum_{j=1}^{n} \lambda_j a_j = b \\
& \Rightarrow \textstyle\sum_{j=1}^{n} \lambda_j a_j = b - \sum_{j=1}^{n} x_j a_j \\
& \Rightarrow \lambda^T A = b - x^T A \\
& \Rightarrow \lambda^T = bA^{-1} - x^T A A^{-1} = bA^{-1} - x^T \\
& = bA^{-1} - (A^{-1}b) = bA^{-1} - b^T (A^{-1})^T = 0
\end{aligned}$$

Hence, $\lambda_1 = \ldots = \lambda_n = 0$, which means that the columns of A are linear independent, i.e. x is a basic solution □

Theorem 13. (Integrality theorem) *For network-flow problems with integer data, every basic feasible solution, and in particular, every basic optimal solution assigns integer flow to every edge.*

Proof. Let G be a graph with a related integer data set. We will show that integers are assigned to the edge flows of every basic feasible solution by considering the following cases:

(i) $|\mathbf{G}| \leq \mathbf{1}$: The basic optimal solution for both the empty graph and the graph with only one edge and integer data , must obviously assign integer flow to its (nonexisting) edge.

(ii) **G contains a cycle:** We start by randomly selecting an edge for which we assign integer value. Then, it follows that the rest of the edges in the graph must also have integer values because the graph has an integer data set.

(iii) **G is a forest:** Then, G must have at least one leaf node, and by selecting the edge from this vertex, the corresponding value will be given. This edge value must be an integer since the graph has integer data. Then, we continue with the adjacent vertex and observe that the rest of the edges also must have integer edge flows.

(iv) **G contains a matching:** In the matching, the edges corresponding to matched elements will equal 1 by the definition of matchings, i.e. integer value.

Since we will have integer valued edge flows for all possible variants of the graph, we conclude that network-flow problems with integer data must indeed have an integer basic optimal solution. □

With these theoretical results in mind, we can proceed to the actual second proof of the Birkhoff and von Neumann theorem.

Proof. In graph notation, Ω_n will be the set of all possible edge flows for a bipartite graph $G = (V, W; E)$ where $|V| = |W| = n$, satisfying the constraints in (4.3). Also, each of the permutation matrices will correspond to a perfect matching. We let the set of all such perfect matchings of G constitute a set M_n. According to the theorem, we must therefore show that

1. the extreme points of Ω_n corresponds to the perfect matchings in M_n
2. Ω_n is the convex hull of M_n

In order to show the first statement, we must show that

(i) all perfect matchings in M_n are extreme points of Ω_n

(ii) all extreme points of Ω_n are perfect matchings in M_n

For the second statement, it is enough to show that the extreme points of Ω_n are the elements of M_n since we know that all polytopes, including Ω_n, are equal to the convex hull of its extreme points. which is in fact what we

will show in (ii). Hence, we must prove (i) and (ii) in order to prove the theorem.

(i) To show that the elements of M_n are the extreme points of Ω_n, we assume for contradiction that there exists a perfect matching $M \in M_n$ that is not an extreme point of Ω_n. Then, M can be written as the midpoint of some $X_1, X_2 \in \Omega_n$, i.e.

$$M = \frac{1}{2}X_1 + \frac{1}{2}X_2$$

We know that since M is a perfect matching, all edges $(i,j) \in M$ must have edge flow $m_{ij} = 1$. If we let x^1_{ij}, x^2_{ij} be the corresponding edge flows for the edge flow sets X_1 and X_2 respectively, we get

$$1 = m_{ij} = \frac{1}{2}x^1_{ij} + \frac{1}{2}x^2_{ij} = \frac{1}{2}(x^1_{ij} + x^2_{ij})$$

Hence,

$$x^1_{ij} + x^2_{ij} = 2$$

Since $0 \leq x_{ij} \leq 1$ for all edge flows of any edge flow set $X \in \Omega_n$, we must have $x^1_{ij} = x^2_{ij} = 1$. Therefore, $X_1 = X_2$, which means that M cannot be written as a midpoint of two other elements of Ω_n and must therefore be extreme.

(ii) In order to prove that all extreme points of Ω_n are elements of M_n, we will use proposition 12 and theorem 13 presented above. By the proposition we know that all extreme points of Ω_n will be basic solutions, which by the integrality theorem will be integer solutions whenever we have integer data, which we have by the definition of Ω_n (4.3). Hence, the extreme points of Ω_n have integer value and correspond to perfect matchings, i.e all extreme points of Ω_n are perfect matchings, hence elements of M_n.

Alternatively, we can prove (ii) by considering graph properties only.

(ii) We will prove that all extreme points of Ω_n are perfect matchings by assuming for contradiction that there exists an extreme point, X, that is not a perfect matching. Then, this edge set must have at least one edge, (i,j), with corresponding edge flow $0 < x_{ij} < 1$. This results in that there also must exist edges (i,l) and (k,j) where $l \neq j$ and $k \neq i$ with corresponding edge flows $0 < x_{il}, x_{kj} < 1$. By continuing this way, we will eventually obtain an even cycle with edges $(i,j),(j,k),\ldots,(l,i)$ and corresponding edge flows $0 < x_{ij} < 1$ for all i,j. We then define a new graph, $H = (V,W;F)$, with edges corresponding to the edges in E and an edge flow set Y where $y_{ij} = 1$ if (i,j) is directed forwards and $y_{ij} = -1$ if (i,j) is directed backwards. Then,

we can find edge flows $X_1, X_2 \in \Omega_n$ defined as $X_1 = X + \epsilon Y$ and $X_2 = X - \epsilon Y$ for some $\epsilon > 0$. By combining these equations, we can write

$$X = \frac{1}{2}X_1 + \frac{1}{2}X_2$$

Hence, X is not extreme which contradicts our assumption and therefore, X must be a perfect matching. □

Chapter 6

An LP Problem

In chapter 4 we presented the assignment problem, and observed that it was an LP problem having the extreme points of Ω_n as its solution space. In this chapter we will present another problem having elements of Ω_n as its possible solutions, but not necessarily the entire Ω_n as its solution space. We will show that this problem also is an LP problem which can be written on the form (2.2). Therefore, we can use the Simplex method in order to solve some examples. Further, we will relate such problems to tridiagonal matrices, which will be presented based on the theory in Dahl's article[4] and give rise to a more rigorous version of this problem. Also, a more liberal variation of this problem will be presented and exemplified and will finish off this chapter.

6.1 The Problem

Let n be a positive integer, and let S and N be to disjunct subsets of $Q = \{(i,j) : 1 \leq i,j \leq n\}$ (where $S \neq \emptyset$). Q is now the set of all positions in an $n \times n$ matrix. Consider the problem:

$$\gamma(S,N) = \sup\{ \sum_{(i,j)\in S} a_{ij} : A = [a_{ij}] \in \Omega_n, a_{ij} = 0 \text{ for } (i,j) \in N\} \qquad (6.1)$$

In this problem, we are given sets S and N as positions of a doubly stochastic $n \times n$ matrix A, and must find the A related to the maximal value of γ satisfying the constraints these sets apply. We can consider this an LP problem on the form stated in (2.2) by letting the entries of A be our decision variables, i.e. A corresponds to X in (2.2) and must therefore be doubly stochastic. We present the sets S and N as $n \times n$ matrices S and N where $n_{ij} = 1$ and $s_{ij} = 1$ if $(i,j) \in S$, $n_{ij} = 0$ and $s_{ij} = 0$ if $(i.j) \notin S$, and finally if

$(i,j) \notin N, S$ we let $n_{ij} = 1$ and $s_{ij} = 0$. Because we want to sum the entries of S only when deciding γ, we must force a_{ij} to be zero when $(i,j) \in N$. By this we can write our problem in the following way

$$\begin{array}{lll} \text{Maximize} & \sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}a_{ij} & \\ \text{Subject to} & a_{ij} \leq n_{ij} & \forall\, i,j = 1,\ldots,n \\ & \sum_{i=1}^{n} a_{ij} = 1 & \forall\, j = 1,\ldots,n \\ & \sum_{j=1}^{n} a_{ij} = 1 & \forall\, i = 1,\ldots,n \\ & a_{ij} \geq 0 & \forall\, i,j = 1,\ldots,n \end{array} \tag{6.2}$$

or equivalently in matrix notation

$$\begin{array}{ll} \text{Maximize} & \langle S, A\rangle \\ \text{Subject to} & A \leq N \\ & eA = e \\ & Ae^T = e^T \\ & A \geq 0 \end{array} \tag{6.3}$$

where $e = (1,1,\ldots,1)$. We observe that the constraints of (6.2) and (6.3) consist of $2n^2$ inequalities and $2n$ equalities. And by the inequalities in the last line, the decision variables must nonnegative. Hence, (6.2) and (6.3), and therefore also the problem stated in (6.1) are LP problems.

Now that we have confirmed that (6.1) is an LP problem, we can use the Simplex method in order to solve some examples, where we let S and N vary, by implementing these examples in an OPL-CPLEX program. The program code will be in the appendix.

Example 10. *We choose different matrices S and N, and obtain γ and the corresponding solution matrix A.*

1.

$$S = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \text{ and } N = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \Rightarrow A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

with optimal value $\gamma = 3$.

2.

$$S = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \text{ and } N = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \Rightarrow A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

with optimal value $\gamma = 2$

3.

$$S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{ and } N = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \Rightarrow A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

with optimal value $\gamma = 3$.

In these examples, we have $Q = N \cup S$ and therefore the matrices S and N are equal. But we could also have elements of Q for which $(i, j) \notin N, S$. Then, S and N will be represented in matrices that will not look the same, which is why the sets S and N are presented in different matrices.

Some possible solution matrices, A, for different choices of S and N were presented in the examples above. Now, we will take a closer look on what kind of solution matrices that will constitute the feasible solutions of a given problem, and which of these matrices that will be optimal solutions. First, we need to determine what values the optimal solution can have. In the 3×3 problems above, we observed that γ never exceeded 3. In general for an $n \times n$ problem, we therefore claim that $\gamma \leq n$ because each column sum and each row sum for matrices, A, representing a solution, must equal 1 by definition. Hence, the sum of all entries of A must equal the sum of all column sums, which is equal to the sum of all row sums, which again is equal to n. Since we are summing the entries in S only, this value will work as an upper bound, i.e. $\gamma \leq n$.

We now proceed by examining how different choices of the sets N and S will influence A. When $S = Q$, which means that $N = \emptyset$, any entry of A can have nonzero value, and therefore, A can be any matrix in Ω_n. Since we are summarizing all $(i, j) \in Q$, we will always get $\gamma = n$. Hence, all elements of Ω_n will be both feasible and optimal solutions. If we instead let $S \in \{P_n\}$ and N be any set in $\{Q \backslash S\}$, then as long as $a_{ij} = 0$ if $(i, j) \in N$ all $A \in \Omega_n$ will be feasible. For $N = \emptyset$, all $A \in \Omega_n$ will be feasible as well, but unlike in the previous case, only $A = S$ will be optima,l since this is the only way of obtaining $\gamma = n$. If we expand N such that $S \cup N = Q$, we observe that $A \in P_n$ will be the only feasible solution matrices. Thus, all feasible solutions

will also be optimal with $\gamma = n$ as optimal value.

For the choices of S and N we now have presented, there will always exist feasible solutions. Generally speaking, this is not always the case. In order for the solution matrix, A, to be feasible, it must be possible to assign nonzero elements to at least one position in each row and each column. Otherwise, the row sums and column sums cannot possible equal 1, which means that $A \notin \Omega_n$. Also, in order to obtain $\gamma = n$, S must contain at least n elements, since each element adds at most one to the sum. So in general

> For a feasible A to exist: $\{Q \backslash N\}$ must contain at least one element in each row and each column
>
> For optimal value $\gamma = n$ to be possible: S must contain at least one element in each row and each column

By this we conclude that we obtain the smallest sets, S satisfying $\gamma = n$, when $|S| = n$, i.e. $S \in P_n$, which means that we have $n!$ such sets.

By König's theorem we can also decide how big γ can possibly be by observing the positions of $S \subseteq Q$. Because $\max|M|$ denotes the maximum number of matched elements a graph can possibly have, i.e. the maximum number of entries that can equal one in a doubly stochastic matrix, we have $\max|M| = \gamma$. Then, because $\max|M| = \min|C|$, we can easily determine γ for given problems by examining the corresponding matrix S, and count the number of lines necessary in order to cover all the elements equal one.

This far, our LP problem has lacked an actual cost matrix and has only had possible and impossible positions of nonzero elements in A. We will soon return to this problem with an added cost matrix, but first we will present a new subclass of matrices which will add some extra restrictions to our LP problem.

6.2 Tridiagonal Matrices

The tridiagonal matrices are characterized by having entries only on the diagonal, the subdiagonal and the superdiagonal. Since such matrices are contained in Ω_n, all row sums and column sums must equal one. For the problem (6.1), this corresponds to N consisting of all positions $(i, j) \in Q$ that are not on the diagonals. We let all matrices satisfying these properties constitute a polytope, Ω_n^t, which has a characteristic structure. We define the polytope

$$P_n^\mu = \{\mu \in \mathbb{R}^{n-1} : \mu \geq 0, \mu_i + \mu_{i+1} \leq 1 \, (1 \leq i \leq n-2)\} \tag{6.4}$$

for $n \geq 3$ to consist of vectors in $\mathbb{R}^{n-1}$, each defining a tridiagonal matrix. For n=2, we let $P_2^\mu = [0, 1]$. Then, we can define the elements of Ω_n^t to be on the form

$$A_\mu = \begin{pmatrix} 1-\mu_1 & \mu_1 & 0 & \dots & & & 0 \\ \mu_1 & 1-\mu_1-\mu_2 & \mu_2 & \dots & & & 0 \\ 0 & \mu_2 & 1-\mu_2-\mu_3 & \dots & & & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \mu_{n-2} & 1-\mu_{n-2}-\mu_{n-1} & \mu_{n-1} \\ 0 & 0 & 0 & \dots & 0 & \mu_{n-1} & 1-\mu_{n-1} \end{pmatrix} \tag{6.5}$$

where each vector $\mu \in P_n^\mu$ defines a specific matrix A_μ, meaning that Ω_n^t and P_n^μ are affinely isomorphic.

Proposition 14. $\Omega_n^t = \{A_\mu : \mu \in P_n^\mu\}$

We can easily observe that these matrices are symmetric, having μ on both the subdiagonal and superdiagonal. By letting f_n denote the nth Fibonacci number, the following theorem will yield some further properties of Ω_n^t.

Theorem 15.

(i) *Ω_n^t is a polytope in $\mathbb{R}^{n\times n}$ of dimension $n-1$ with f_{n+1} vertices.*

(ii) *The vertex set of Ω_n^t consists of all tridiagonal permutation matrices; these are matrices of order n and can be written as a direct sum*

$$A = A_1 \oplus A_2 \oplus \ldots \oplus A_t \tag{6.6}$$

where each

$$A_i = \begin{cases} K = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \\ J = [1]. \end{cases}$$

(iii) *Consider a vertex A. Then each adjacent vertex of A is obtained from A by either*

- *(a) interchanging a sequence of consecutive blocks J, K,..., K (with t K's) and a sequence K, K,..., K, J (with t K's)*
- *(b) interchanging a sequence of consecutive blocks K, K,..., K (with t K's) and a sequence J, K,..., K, J (with $t-1$ K's)*

Proof. For a proof of this theorem, the reader is referred to Dahl's article [4]. □

6.3 Applications

Our LP problem (6.1) can be applied to all situations that can be expressed as a bipartite graph, $G = (V, W; E)$, where not all connections between the vertex sets V and W are allowed or possible. The previous mentioned transportation problem can be considered such an LP problem and can be written

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = a_i \quad \text{for } i = 1, \ldots, n \\ & \sum_{i=1}^{n} x_{ij} = b_j \quad \text{for } j = 1, \ldots, n \\ & x_{ij} \geq 0 \qquad \text{for } i, j = 1, \ldots, n \end{aligned}$$

in matrix notation where the elements, x_{ij}, will state how many units that should be transported from factory i to customer j, and c_{ij} is the unit cost related to transporting items from factory i to customer j. We say that factory i produces a_i units of this item, and that customer j wants to buy b_j items in total. As before, we relate this to (6.1) by letting X correspond to the solution matrix A. Hence, X must be doubly stochastic, which means that we must have $a_i = b_j = 1 \; \forall \; i, j$. We will therefore interpret the elements a_{ij} of A to be the percentages of the total amount of units transported/received by the factories/customers.

Usually, it will be most convenient for factories to transport their items to customers within a certain distance only. If a distance boundary of the transportations is given, certain factory-customer relations, (i, j), will not be allowed, i.e. $(i, j) \in N$. By this, we can define N and S according to certain transportation limits and obtain an optimal soultion (if it exists) satisfying this. If we want each factory to transport items to the customers in its own and its neighbouring towns only, then A must be tridiagonal and will be on the form stated in (6.5) for a given μ. In problem (6.1) we are maximizing the sum of the entries in matrix A without any cost matrix. This differs from the transportation problem which first of all is a minimization problem, and secondly has a cost matrix. To overcome these differences, we first use the fact that

$$\max \langle S, A \rangle = -\min\langle -S, A \rangle$$

By letting S be a matrix with elements $s^*_{ij} = -s_{ij}$, we get the following maximization problem

$$-\text{Maximize } \langle S, A \rangle$$

with the same constraints as before. Then, we can transform this transportation problem into a problem on the form (6.1) and compute

$$\gamma(S,N) = -\sup\{\sum_{(i,j)\in S} s^*_{ij}a_{ij} : A = [a_{ij}] \in \Omega^t_n, a_{ij} = 0 \text{ for} (i,j) \in N\} \quad (6.7)$$

Still, this problem will not have any actual cost matrix since S is only containing elements equal -1 or 0, which will only state whether a transportation is possible or not. Therefore, all possible transportation distances will be equally likely.

Example 11. *In a certain area, there is 6 towns, each containing one factory and one customer. These towns are positioned along a road where Town 1 is the first town and Town 6 is the last town. In order to limit the transportation distances, each factory will only transport items to the customer in its own and its neighbouring towns. The transportation costs are all the same, which means that all transportations to any possible location are equally likely. Since a transportation distance is either allowed or not allowed,* $S \cup N = Q$ *resulting in the following matrix representations of* S *and* N

$$S = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \text{ and } N = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

By using the OPL-CPLEX program to solve this problem, we get

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

which means that the factory in Town 1 transports all its items to the customer in Town 2, the factory in Town 2 transports all its items to the customer in Town 1, the factory in Town 3 transports all its items to the customer in Town 4, the factory in Town 4 transports all its items to the customer in Town 3, the factory in Town 5 transports all its items to the customer in Town 6 and finally, the factory in Town 6 transports all its items to the customer in Town 5. Because A *is on the form (6.6), it is an extreme point of* Ω^t_n.

We observe that for problems on the form (6.7), all matrices $A \in \Omega_n^t$ will be both feasible and optimal solutions. By adding a cost matrix, however, we will no longer have optimality for every feasible solution, and the problems will be more relatable to the real world. We do this by simply transforming the -1 entries of our matrix S into entries that will represent the given costs of a problem. Then,

$$\gamma(S, N) = -\sup\{\sum_{(i,j)\in S} s_{ij}^* a_{ij} : A = [a_{ij}] \in \Omega_n^t, a_{ij} = 0 \text{ for } (i,j) \in N\} \quad (6.8)$$

will be the new transportation problem, where s_{ij} now denotes the cost of transporting items from factory i to customer j, which might be related to transportation distance, transportation time, certain local decisions or a combination. We assume that the smallest transportation cost from a certain factory is not necessarily to the customer in the same town. Otherwise, the solution would be trivial and correspond to the identity matrix. We will now resume the previous example.

Example 12. *We add costs to the possible transportation distances such that*

$$S = \begin{pmatrix} 60 & 50 & 0 & 0 & 0 & 0 \\ 45 & 55 & 35 & 0 & 0 & 0 \\ 0 & 50 & 60 & 45 & 0 & 0 \\ 0 & 0 & 30 & 45 & 40 & 0 \\ 0 & 0 & 0 & 55 & 50 & 65 \\ 0 & 0 & 0 & 0 & 45 & 60 \end{pmatrix}$$

Then, we can use the same OPL-CPLEX program as before in order to solve this new problem, which gives us the solution matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

with the corresponding optimal value $\gamma = 280$.

6.4 A relaxation of the Linear Programming problem

Because of the strict constraints that we apply to problems on the form (6.1) given by N and the fact that we must have $A \in \Omega_n$, these problems will not

necessarily have any feasible solution. As we have already observed, we must be able to assign nonzero elements to at least one entry in each row and each column of A where $a_{ij} \leq n_{ij}$ is satisfied, in order to obtain a feasible solution. However, if we relax the constraints

$$\begin{aligned} \sum_{j=1}^{n} a_{ij} &= 1 \quad i = 1, \ldots, n \\ \sum_{i=1}^{n} a_{ij} &= 1 \quad j = 1, \ldots, n \end{aligned}$$

to be

$$\begin{aligned} \sum_{j=1}^{n} a_{ij} &\leq 1 \quad i = 1, \ldots, n \\ \sum_{i=1}^{n} a_{ij} &\leq 1 \quad j = 1, \ldots, n \end{aligned}$$

there will always exist a feasible solution to our problem no matter what N is. Our relaxed problem with network-flow notation will then be

$$\begin{aligned} &\text{Maximize} && \sum_{e \in E} s_e a_e && \\ &\text{Such that} && \sum_{e \in \delta(v)} x_e \leq 1 && \forall\, v \in V \cup W \\ & && x_e \geq 0 && \forall\, e \in E \end{aligned} \tag{6.9}$$

We define $\chi^M(e) \in \mathbb{R}^{|E|}$

$$\chi^M(e) = \begin{cases} 1 & \text{if } e \in M \\ 0 & \text{otherwise} \end{cases}$$

to be the characteristic vector for some matching $M \subseteq E$ for a bipartite graph $G = (V, W; E)$. Then, we can write the solution space for (6.9) as

$$M_G = \{x \in \mathbb{R}^{|E|} : \sum_{e \in \delta(v)} x_e \leq 1 \; \forall\, v,\; x_e \geq 0\}$$

We observe that the extreme points of this set are the set all perfect matchings, hence

$$M_G = \text{conv}(\chi^M : \text{M is a perfect matching in G})$$

which is the matching polytope.

Example 13. *We will now present a simple application of (6.9) inspired by Hall's Marriage theorem. We consider a group of n women and n men. We*

want each man and woman to write a prioritized list of possible spouses by assigning a value between 0 and 100 to each name on their list according to how much happiness they relate to a marriage to this person. 100 denotes perfect happiness and 0 means that a marriage to this person is out of the question. They are not obliged to assign nonzero values to n names on their list, in fact if they don't think that anyone in the group will be suitable, they can assign zero to all names. We arrange these $2n$ *lists into an* $n \times n$ *cost matrix* S *where* s_{ij} *denotes the happiness the ith women relates to a marriage to the jth man multiplied by the happiness the jth man relates to a marriage to the ith woman, i.e.* $0 \leq s_{ij} \leq 10000$*. If this number is zero, such marriage is impossible and hence* $(i, j) \in N$*. For our solution matrix,* A*, we let* $a_{ij} = 1$ *if the ith woman marries the jth man, and zero otherwise. Therefore, the problem in this example will be on the form (6.9) as a relaxed assignment problem. Hence, the possible solutions will be among the extreme points of* M_G*.*

We want to maximizes the total happiness corresponding to as many marriages as possible, i.e. this is a matchmaker problem. Because the members of our group have the option of blacklisting certain names, we will not necessarily get n marriages. E.g. if a woman has only one man on her list, but this man relates more happiness to a marriage to another woman, there is a good chance that he will end up with her instead, whereas the woman ends up alone. For the relaxed version of our LP problem, this is allowed, i.e. we allow spinsters, old maids, etc.

Even though this example is not very applicable to the real world, it states the basic properties of such problems and the solutions we can get. We will end this chapter by presenting an example that is more relatable to our world.

Example 14. *We consider a group of n people where each member has a certain amount of capital that they want to invest in some companies. For simplicity, we let it be n companies in total. The variables,* a_{ij}*, in our matrix,* A*, will denote the percentage of the ith persons capital that is invested in the jth company. Each person is not obliged to invest all of his or her capital, hence* $\sum_{j=1}^{n} a_{ij} \leq 1$*. Because each company has a limited amount of shares, we assume* $\sum_{i=1}^{n} a_{ij} \leq 1$ *as well. This gives us a relaxed LP probem, which can be written on the form (6.9). We also add a cost matrix,* S*, to this problem, where* s_{ij} *denotes how much person i gets out of investing all of his or her capital in company* j*. For each person we introduce a lower bound,* b_i*, such that if the profit of investing in a company is lower than this value, i.e. if* $s_{ij} < b_i$*, any investment in this company for this person is ruled out,*

hence $(i, j) \in N$. *We also let* $(i, j) \in N$ *if the ith person doesn't want to invest in company* j *of personal reasons etc. With all these preliminaries determined, we can find the solution of such problems by maximizing* $s_{ij}a_{ij}$, *which will give us the invesiting pattern for which our group will achieve the highest total profit possible (assuming that the group have a very collectivist mindset).*

Chapter 7

Assignment Problems in a High School Setting

The theoretical background, results, applications and algorithms of the assignment problem and doubly stochastic matrices have been the main focus of this thesis so far. But since this is in fact also a part of my teacher education, I will conclude by a didactical discussion about whether and how to implement these concepts to a school setting. Because my education is mainly directed towards high school level and this will also be the level where the students will be most receptive for such theory, this level will be the main focus of this chapter. I will start by relating the assignment problem to the curriculum in the existing math courses and continue by discussing how it may be appropriate to introduce this problem according to the existing curriculum and the students prerequisites and knowledge. Finally, I will argue why such implementation could be useful.

In the current math curriculum, optimization is actually a part of the second year high school course S1. In the rest of the math courses, optimization is nonexisting. In S1, the LP problem is introduced and the students learn how to solve such problems graphically by testing different linear functions and selecting the function corresponding to the maximum value of the problem within a certain area of values. Because matrices are not part of the high school curriculum, the LP problem is presented as a set of linear equalities and inequalities in two variables. Also, the lack of matrix knowledge makes it impossible to introduce the Simplex method in order to solve these problems, which in my opinion would have been a more mathematical approach than the existing trial and error method.

In this thesis, we have presented the assignment problem in both matrix and graph notation. We will now discuss whether either of these approaches could

be suitable in order for high school students to understand the assignment problem. By introducing this problem in matrix notation, the students must necessarily first be familiar with matrix concept. Basically, a matrix is just a system of equations of several variables, which are introduced as a concept even before the students are starting high school. Also, vectors, which actually are $m \times 1$ matrices, are part of some of the math courses in high school. On the other hand, matrices have also many similarities with tables, which the students are highly familiar with. Therefore, the matrix concept is not that distant from the existing curriculum and the students prerequisites, and could beneficially be introduced as a general concept in the first year of high school. This would give the students a better foundation in order to learn both the optimization theory and the concept of vectors later on. With the matrix concept established, doubly stochastic matrices could easily be introduced and then also the assignment problem. Even though the students lack the theoretical foundation necessary in order to achieve a complete understanding of this problem, its real world applicable nature makes it more meaningful and interesting to present actual problems instead of presenting the underlying theory. As we know, the Hungarian method solves the assignment problem with a matrix approach, and when it is presented as a step by step recipe procedure, it could be a both comprehensible and interesting algorithm for the students to learn and play around with in order to solve actual problems.

The assignment problem could also be presented in graph notation. Even though such theory differs from the existing curriculum in many ways, the concept itself is quite simple. The box and line patterns might actually resemble the sort of activities the students did in their primary school years. As in the matrix approach, the underlying theory will probably be too complex for most high school students to grasp, but since the assignment problem is highly applicable to real world settings even without this knowledge, the actual problem will be understandable. In the simplest way, it can be introduced as a bipartite graph with two sets of boxes connected by some lines with different associated values, for which the students are supposed to find a pattern maximizing or minimizing the corresponding total value according to some restrictions given by the assignment problem. In order to avoid trial and error as the solution method of this problem, the shortest augmenting path algorithm presented earlier can be introduced as a step by step algorithm. As long as the students can relate the given problems to their own world, I believe that this will be manageable for most students.

Clearly, implementing the assignment problem into the high school math curriculum is possible either with a matrix or a graph approach. However,

will such implementation be useful? As stated earlier, such problems are applicable to the real world in numerous ways and could therefore be varied according to the students interests. Also, when presented properly, the underlying mathematical concepts will be quite comprehensible for high school students. The real world applicability property is often lacking in the existing curriculum, and consequently the students are asking themselves (and the teacher) "why do we learn this? I will never use this in the real world." The last statement is usually not true, but because the students are not able to relate the concepts to his or her reality, it will seem that way. Therefore, it could be motivating to learn some math where this is not the case. Such experiences can also result in an increased interest in math in general. So even though this theory is quite untraditional compared to the current curriculum, we know that the world has changed a lot over the past 100 years, maybe it is time for high school math curriculum to do this as well.

Appendix A

Program codes

A.1 The Hungarian Algorithm - Matlab code

```
function [X] = hungarian(C,n)
[C]=part1(C,n);
[X,C]=part2(C,n);
z=0;
for i=1:n
    for j=1:n
        if X(i,j)==1
            z=z+1;
        end
    end
end
while z<n
    [C]=part3(C,X,n)
    [X,C]=part2(C,n)
    z=0;
for i=1:n
    for j=1:n
        if X(i,j)==1
            z=z+1;
        end
    end
end
end
end
function[C2]=part1(C,n)
    % Row reduction
    for i=1:n
        u(i)=10000000;
        for j =1:n
```

```
            if C(i,j)<u(i)
                u(i)=C(i,j);
            end
        end
    end
    for i=1:n
        for j=1:n
            C1(i,j)=C(i,j)-u(i);
        end
    end
    % Column reduction
    for j=1:n
        v(j)=100000;
        for i=1:n
            if C1(i,j)<v(j)
                v(j)=C1(i,j);
            end
        end
    end
    for j=1:n
        for i=1:n
            C2(i,j)=C1(i,j)-v(j);
        end
    end
end
function[X,C2]=part2(C2,n)
X=zeros(n,n);
    b=0;
    e=0;
    B=zeros(n,n);
    % Assigning rows
    for i=1:n
        for j=1:n
            if C2(i,j)==0
                B(i,j)=0;
            else
                B(i,j)=1;
                e=e+1; %number of entries in X not assignable
            end
        end
    end
    q=0;
    while e<n*n %as long as we have assignable elements
    for i=1:n
        k=0;
        a=0;
        for j=1:n
            if B(i,j)==0
                k=k+1; %number of zeros in row
```

```
            a=j;
        end
    end
    if k==1
        X(i,a)=1; %make assignment
        b=b+1;
        for l=1:n
            B(l,a)=1; %one less assignable entry
        end
    end
end
% Assigning columns
if b<n
    for j=1:n
        d=0;
        c=0;
        for i=1:n
            if B(i,j)==0
            d=d+1; %number of zeros in column
            c=i;
            end
        end
    if d==1
        X(c,j)=1; %make assignment
        b=b+1;
        for f=1:n
            B(c,f)=1;%one less assingable entry
        end
    end
    end
end
e=0;
% update number of unassignable elements
for i=1:n
    for j=1:n
        if B(i,j)==1
        e=e+1;
        end
    end
end
q=q+1;
    if q>10
        i=1;
        while i<n
            w=0;
            v=0;
            for j=1:n
                if B(i,j)==0
                    w=w+1;
```

```
                    end
                end
                    if w>1
                        for j=1:n
                            if B(i,j)==0
                                X(i,j)=1;
                                b=b+1;
                                for l=1:n
                                 B(l,j)=1; %one less assignable entry
                                 B(i,l)=1;
                                end
                                i=n;
                                j=n;
                                q=0;
                                w=0;
                            end
                        end
                        i=n;
                    end
            end
        end
    end
end
function[C]=part3(C,X,n)
x=zeros(n,1);
y=zeros(n,1);
    for i=1:n
        u=0;
        for j=1:n
            if X(i,j)==1;
                u=u+1;
            end
        end
        if u==1
            x(i,1)=0;
        else
            x(i,1)=1; % mark unassigned rows
        end
    end
    for i=1:n
        if x(i,1)==1
            for p=1:n
                if C(i,p)==0 % if a marked row has a zero
                    y(p,1)=1; %mark corresponding column
                end
            end
        end
    end
    for j=1:n
```

```
        if y(j,1)==1
            for q=1:n
                if X(q,j)==1 %if a marked column has an assignment
                    x(q,1)=1; %mark corresponding row
                    for k=1:n
                        if C(q,k)==0 %if marked row has a zero
                            y(k,1)=1; %mark corresponding row
                        end
                    end
                end
            end
        end
    end

    x; %marked rows
    y; %marked columns
    delta=100000;
    for i=1:n
        for j=1:n
            if x(i,1)==1 && y(j,1)==0
                    if C(i,j)<delta
                        delta =C(i,j); %smallest number with no line
                    end
            end
        end
    end
    for i=1:n
        for j=1:n
            if x(i,1)==0 && y(j,1)==1
                %add delta to entries with two lines through it
                C(i,j)=C(i,j)+delta;
            else if x(i,1)==1 && y(j,1)==0
                    %subract delta from entries with no lines through it
                    C(i,j)=C(i,j)-delta;
                end
            end
        end
    end
end
```

A.2 The Hungarian Algorithm - OPL-CPLEX program

```
int n =...;

range Rows = 1..n;
```

```
range Cols = 1..n;

float C[Rows][Cols] = ...;
dvar int+ X[Rows][Cols];

 minimize
 sum(i in Rows)sum(j in Cols) C[i][j]*X[i][j];
subject to {
  forall (i in Rows)
    d1: sum(j in Cols) X[i][j]==1;
  forall (j in Cols)
    d2: sum(i in Rows) X[i][j]==1;
}
execute {
  writeln (" Optimal value : " , cplex . getObjValue ( ) ) ;
   for ( var j in Cols ){
   for (var i in Rows)
  {
    write (X[i][j] , " " ) ;
  }
  }
  writeln (" ] ");
}
```

A.3 Problem 6.1 - OPL-CPLEX program

```
int n = ...;
range nodes = 1..n;
range edges = 1..n;
int S[nodes][edges]=...;
int N[nodes][edges]=...;
dvar float+ A[nodes][edges];

maximize
sum(j in edges)sum(i in nodes) S[i][j]*A[i][j];
subject to {
  forall(i in nodes, j in edges)
    d1: A[i][j]<=N[i][j];
  forall (i in nodes)
    d2: sum(j in edges) A[i][j]==1;
  forall (j in edges)
    d3: sum(i in nodes) A[i][j]==1;
}
execute {
  writeln (" Optimal value : " , cplex . getObjValue ( ) ) ;
   for ( var j in edges ){
   for (var i in nodes)
```

```
  {
    write (A[i][j] , " " ) ;
  }
  }
  writeln (" ] ");
}
```

Bibliography

[1] Bellù, L. G., & Liberati, P. (2006). *Inequality Analysis. The Gini Index*. Retrieved from `http://www.fao.org/docs/up/easypol/329/gini_index_040en.pdf`

[2] Bertsekas, D. P. (1988). The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem. *Annals of Operations Research*, 14(1-4), 105-123.

[3] Burkhard, R.E., & Çela, E. (1999). Linear Assignment Problems and Extensions. In D.-Z. Du & P.M. Pardalos (Eds.), *Handbook of Combinatorial Optimization. Supplement Volume A*, 1999. (pp. 75-149). New York: Springer-Verlag.

[4] Dahl, G. (2004). Tridiagonal doubly stochastic matrices. *Linear Algebra and its Applications*, 390, 197.208.

[5] Dahl, G. (2010). *Network flows and combinatorial matrix theory*. Retrieved from University of Oslo: `http://heim.ifi.uio.no/~geird/CombMatrix.pdf`

[6] Dahl, G. (2010). *An Introduction to Convexity*. Retrieved from University of Oslo: `http://heim.ifi.uio.no/~geird/conv.pdf`

[7] Lau, L. C., Rav, R, & Singh, M. (2011). *Iterative Methods in Combinatorial Optimization*. Cambridge: Cambridge University Press.

[8] Luenberger, D. G. (1989). *Linear and Nonlinear Programming*. Massachusetts: Addison-Wesley Publishing Company.

[9] Marshall, A. W., & Olkin, I. (1979). *Inequalities: Theory of Majorization and Its Applications*. San Diego: Academic Press.

[10] Vanderbei, R. (2008). *Foundations and Extensions*. New York: Springer.